Departamento de Ingeniería de Sistemas y Automática

PROYECTO FIN DE CARRERA

INTERFAZ DE TELEOPERACIÓN PARA EL MANIPULADOR MÓVIL MANFRED

Autor: José Luis Ortiz Navarro

Tutor: David Álvarez Sánchez

Leganés, junio de 2015

CONTENIDO

TABLA DE IMÁGENES

Capítulo 1. INTRODUCCIÓN

Este proyecto fin de carrera se enmarca en el proyecto "Técnicas de aprendizaje y planificación de manipulación diestra para manipuladores móviles (Dex-arm)", que se dedica a investigar técnicas de aprendizaje y manipulación de objetos, y a desarrollar un conjunto brazo-mano que intente emular las capacidades de manipulación que ofrece el conjunto humano. Este proyecto se sirve del robot Manfred (Man Friendly), un manipulador móvil dotado del brazo robótico LWR-UC3M-1 con seis grados de libertad.

Este proyecto pretende desarrollar una interfaz de fácil acceso y multiplataforma que permita interactuar de forma rápida y ágil con el robot para facilitar la realización de las pruebas que requiere el desarrollo de algoritmos complejos de alto nivel. Pero antes de comenzar a definir unos objetivos más concretos será necesario realizar un estudio previo sobre estado actual de la plataforma de partida: los robots, o más concretamente, los robots manipuladores móviles. Para comenzar, se estudiará el origen de estos dispositivos y su evolución a lo largo de la historia.

1.1. BREVE HISTORIA DE LA ROBÓTICA

Los primeros autómatas de la historia eran aparatos simples y rudimentarios: en el siglo IV a. C., Arquitas de Tarento, un matemático griego, construyó un ave mecánica llamada "La Paloma" que funcionaba a vapor. Tras él, Herón de Alejandría (10 – 70 d. C.) desarrolló varios dispositivos automáticos accionados por aire, vapor o agua y programables por el usuario.

Avanzando unos cuantos siglos, hasta el siglo XII, se puede apreciar una evolución más notable. El inventor musulmán Al Jazari es un buen ejemplo: construyó varias máquinas automatizadas entre los que había útiles de cocina, máquinas musicales y los primeros robots humanoides, que consistían en cuatro músicos a bordo

de un bote en el lago que entretenían a los invitados de las fiestas reales. El mecanismo que movía estos autómatas y accionaban los instrumentos de percusión eran programables, con lo que los ritmos podían ser fácilmente modificados. A continuación puede verse una representación de este robot, ya no tan rudimentario, creada por el propio inventor:

IMAGEN 1 - ROBOT MUSICAL PROGRAMABLE DE AL-JAZARI

En 1799 nace en Japón Hisashige Tanaka, un famoso ingeniero e inventor que fue conocido como el Edison japonés. Experimentó con la neumática y la hidráulica, utilizando aceite a presión para transmitir movimiento. Con sólo 20 años, construyó varios juguetes mecánicos de una extraordinaria complejidad que realizaban diferentes acciones: disparaban flechas cogiéndolas de una aljaba, eran capaces de escribir, o servían té. A continuación dos imágenes de este último, expuesto en el Museo Británico:

IMAGEN 2 - ROBOT JAPONÉS CAPAZ DE SERVIR TÉ

IMAGEN 3 - INTERIOR DE ROBOT JAPONÉS

En 1937 Griffith P. Taylor “Bill”, terminó el primer robot industrial: la “Grúa”. Estaba accionada por un solo motor eléctrico pero era capaz de apilar bloques de madera. La automatización se conseguía mediante el uso de cinta de papel perforado. El prototipo estaba construido casi en su totalidad con piezas de *Meccano*, en 1997 se construyó una réplica completa.

La revolución industrial, que finalizó a mediados de este siglo, influyó de manera determinante sobre el desarrollo de la robótica. Generalizó el uso de la tecnología en los procesos de producción, lo que hizo aumentar inevitablemente la investigación en robótica industrial. Este factor añadido al uso de la electricidad y a la miniaturización de sus componentes, ha provocado que los robos industriales hayan evolucionado hasta llegar a los complejos y precisos equipos especializados que participan hoy día en las cadenas de montajes y las fábricas, como el de la siguiente imagen.

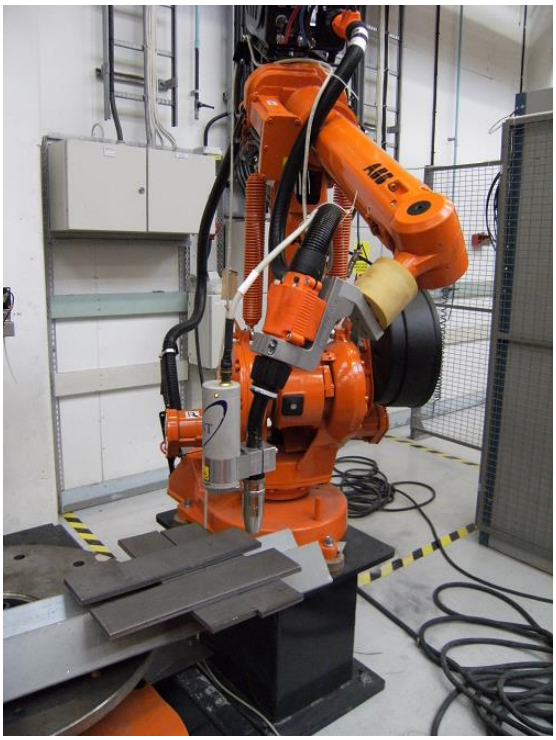


IMAGEN 4 - ROBOT MANIPULADOR MODERNO EN FÁBRICA

Fuera del ámbito industrial, la robótica ha avanzado también en otros sectores de forma notable durante las últimas décadas, como por ejemplo en el mundo militar, un sector que cuenta con una gran inversión. Investigaciones militares han desarrollado robots autónomos muy sofisticados de diferentes tipos: aviones de combate no tripulados (UAV y drones), robots de incursión y transporte que se desplazan con facilidad y autonomía por terrenos difíciles o tele-tanques son sólo algunos ejemplos.

En los últimos años también se ha extendido el uso (y por tanto ha aumentado el desarrollo) de los llamados robots domésticos que nos sustituyen en tareas simples y aburridas. Ejemplos de robots bajo esta clasificación son los aspiradores/limpiadores automáticos, que limpian la suciedad del suelo de cualquier habitación de forma autónoma, o limpia fondos automáticos, que del mismo modo se encargan automáticamente de mantener las paredes y el fondo de las piscinas. También es destacable la evolución que han tenido los robots humanoides. El objetivo de la investigación acerca de este tipo de robots es que sean capaces de replicar a un ser humano en cualquier tarea, aunque actualmente se trata de un objetivo bastante lejano, y es el desarrollo de nuevas tecnologías el que impulsa este campo de la robótica. Empresas como Honda o Sony cuentan con conocidos ejemplos de estos robots (ASIMO y QRIO, respectivamente), pero hay una gran variedad de empresas o universidades desarrollándolos, bien completos o bien alguna de sus partes.

Las imágenes a continuación muestran tres conocidos ejemplos de los tres tipos de robot que se acaban de mencionar: el robot militar 'Cheetah' desarrollado por Boston Dynamics para las fuerzas armadas de Estados Unidos, el robot humanoide ASIMO de Honda y el robot aspirador Roomba, un robot doméstico reciente éxito de ventas (Solomon, 2013).

IMAGEN 5 - ROBOT MILITAR CUADRÚPEDO CHEETAH (BOSTON DYNAMICS)

IMAGEN 6 - ROBOT HUMANOIDE ASIMO (HONDA)

IMAGEN 7 - ROBOT ASPIRADOR ROOMBA (IROBOT)

1.2. MANIPULADORES

Un manipulador es cualquier robot cuya finalidad sea manejar o actuar sobre un material sin que haya contacto directo de éste con el operador.

Originalmente consistían únicamente en brazos robóticos que se crearon para manipular materiales radioactivos o de alto riesgo biológico, aunque con el desarrollo de la robótica ha hecho que sus usos se multipliquen.

Actualmente podemos encontrar robots manipuladores de cirugía asistida capaces de intervenir quirúrgicamente a un paciente sin que el cirujano se encuentre siquiera en la misma habitación o robots de reparación que pueden arreglar pequeños problemas o defectos externos en las naves espaciales sin que los astronautas tengan que salir al exterior, eliminando así los riesgos derivados de hacerlo. Dos ejemplos de estos manipuladores pueden verse en las imágenes a continuación:

IMAGEN 8 - ROBOT DE CIRUGÍA ASISTIDA

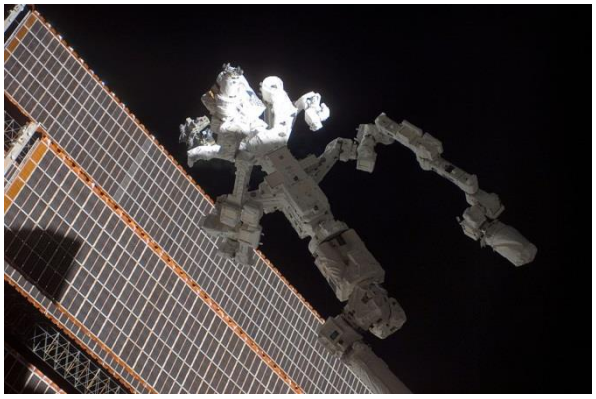


IMAGEN 9 - MANIPULADOR ESPACIAL DEXTRE EN LA EEI

Los manipuladores también se usan porque son capaces de mover grandes y pesadas cargas (por ejemplo, los robots que trasladan y rotan carrocerías completas de coche en las cadenas de montaje) y porque poseen más precisión y velocidad que la que puede alcanzar un humano (como los robots soldadores de micro-componentes electrónicos). Además también son capaces trabajar en espacios reducidos o de difícil acceso (como los ya nombrados robots reparadores).

Los robots manipuladores están definidos por una serie de parámetros, que son comunes a todos ellos:

- Grados de libertad: dos son necesarios para alcanzar cualquier punto en un plano, tres para alcanzar cualquier punto en el espacio. Para controlar completamente la orientación de la punta del robot serán necesarios seis.
- Número de ejes: normalmente coincide con el número de ejes del robot, aunque puede ser superior en caso de que haya ejes no actuados, o subactuados.
- Área de trabajo: región del espacio que el robot puede alcanzar.
- Cinemática del robot: es la relación entre los miembros rígidos del robot y sus articulaciones (la capacidad de giro de éstas), que limita la capacidad de movimientos de éste.
- Capacidad de carga: carga máxima que el robot puede manejar. Habitualmente este parámetro va relacionado con un valor de precisión.
- Velocidad y aceleración máximas para cada eje, o del conjunto.
- Precisión: relación estadística entre la posición deseada y alcanzada. No se trata de un valor fijo, a mayores cargas o velocidades de movimiento la precisión puede disminuir.
- Repetitividad: no debe confundirse con la precisión. La repetitividad tiene que ver con la desviación típica de las muestras de posición tomadas tras varias ejecuciones de un mismo programa, y habitualmente se tiene más en cuenta que la precisión, ya que los problemas derivados de ésta se pueden corregir mediante procesos de calibración.

La forma de controlar estos robots es, típicamente, a través de un ordenador, ya sea conectado directamente o a través de una red, y un software de programación específico de cada robot. También hay equipos que se programan a través de un terminal propio que suele consistir en una pantalla con un teclado especial o una pantalla táctil.

Generalmente el programa de un brazo robótico consiste en un recorrido o trayectoria que pasa por diferentes puntos (que no tienen por qué ser fijos) a una velocidad definida. Al programar el robot esta trayectoria se puede definir por las coordenadas de estos puntos, o bien, las condiciones para alcanzarlos (por ejemplo tocar una superficie), introduciéndolas en el software de programación del robot de la forma y en el lenguaje que corresponda. Lo más común es que a la vez que se van añadiendo puntos a la trayectoria, el software vaya ejecutándolos para comprobar que

se realiza de forma correcta. También existe la posibilidad de prever el movimiento resultante del programa en una simulación, creada por el propio software u otro programa ajeno (ver Imagen 10), de manera que este puede depurarse antes de ejecutarse en el robot.

También existe el método de aprendizaje "*Programming by Demonstration*" (programación por demostración). Para añadir un punto mediante este procedimiento, el software del robot libera los motores de las articulaciones dejándolas sueltas, o bien en un estado en el que mantienen su posición pero que les permite moverse fácilmente cuando se les aplica una fuerza externa. De cualquiera de las dos maneras, el operador puede colocar a mano el brazo en la posición deseada. Cuando ésta se alcanza, el software memoriza la posición del brazo como otro punto en el programa.

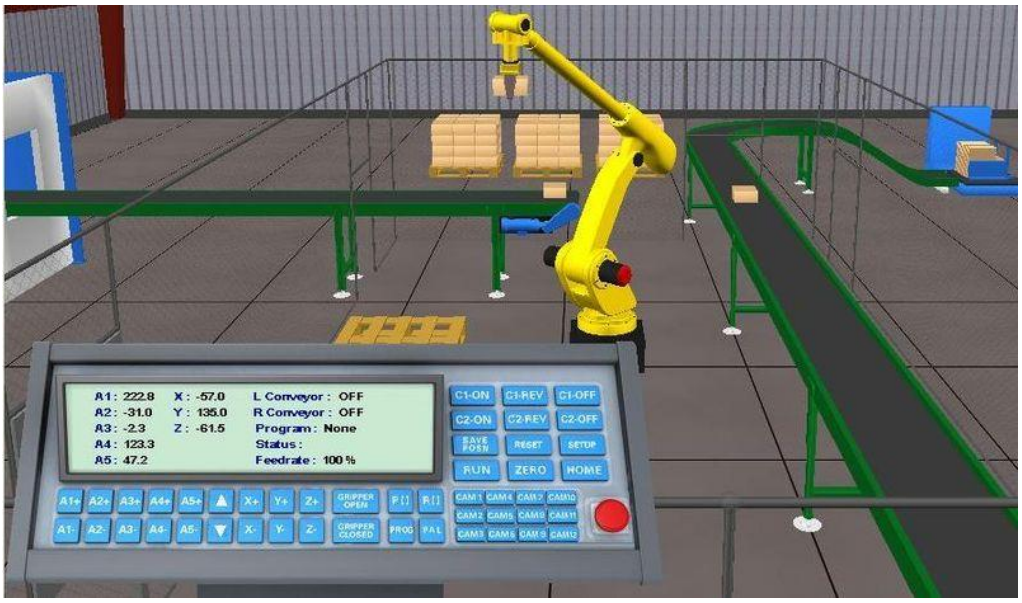


IMAGEN 10 - SIMULADOR DE ROBÓTICA ROBOLOGIX

1.3. ROBÓTICA MÓVIL Y TELEOPERACIÓN

Un robot móvil es aquel que tiene capacidad de locomoción, es decir, que es no está fijado a un punto, sino que es capaz de moverse en su entorno (no tiene por qué hacerlo de forma autónoma). No se incluyen en esta clasificación robots, que si bien no están anclados a un punto fijo y se pueden trasladar fácilmente, no tienen los medios para hacerlo, como algunos robots de cadena de producción que son fácilmente intercambiables entre anclajes o bases en distintas localizaciones.

Los robots móviles son un importante objeto de investigación por parte de universidades, la mayoría de ellas tienen al menos un laboratorio que investiga sobre la materia. Podemos encontrar robots móviles en diferentes ámbitos, como por ejemplo:

- Militar: la locomoción es imprescindible en robots que se utilizan para transportar material por terrenos difíciles o para incursiones, como los tele-tanques o los nuevos robots cuadrúpedos (ver Imagen 5)
- Industrial: robots limpiadores industriales automáticos, o robots de transporte, que mueven cargas de unos puntos a otros pre-programados, incluso de forma totalmente autónoma, gracias a las tecnologías de navegación, tema que se tratará a continuación.
- Doméstico: los ya mencionados robots barredores, robots jardineros o limpia fondos automáticos.

Los robots móviles pueden ser catalogados en distintas clasificaciones, dependiendo de característica que se observe para ello. A continuación se desarrollan dos ejemplos:

- Atendiendo al entorno por el que se mueven podemos encontrar:
  - Terrestres (si se mueven de forma autónoma se les llama UGV, *Unmanned Ground Vehicle*, vehículo de tierra no tripulado)
  - Aéreos (de la misma forma, si son automáticos se les llama UAV, *Unmanned Aerial Vehicle*, vehículo aéreo no tripulado)
  - Submarinos (de nuevo, si no son externamente dirigidos se denominan AUV, *Autonomous Underwater Vehicle*, vehículo autónomo submarino)
- Según su medio de locomoción:
  - De patas, como las de los animales o como las piernas humanas.
  - De ruedas.
  - De oruga, también llamados de vía continua.
- Según su grado de autonomía:
  - Autónomos
  - Teleoperados

En el caso de los robots autónomos, la forma en que éstos planean y deciden su trayectoria forma una parte esencial de su comportamiento: la navegación. Es tan necesario evitar colisiones o situaciones peligrosas como alcanzar los puntos necesarios para ejecutar la funcionalidad correspondiente. Para conseguir esto hay varios mecanismos de navegación robótica, en función de la tecnología utilizada. Un ejemplo es el método sigue líneas: se trata de un funcionamiento exclusivo de los AGVs bastante simple, aunque puede hacerse más complejo ampliando funcionalidades como pueden ser la detección de obstáculos o ajustes dinámicos de velocidad. El movimiento se ejecuta siguiendo un camino prefijado normalmente por una línea visual en el suelo, o un cable electrificado. En su modo más simple, el

algoritmo del movimiento consiste en “mantener la línea en el centro del sensor”, aunque como ya se ha comentado puede ser más complejo. Otro ejemplo de método sencillo de navegación es el de movimiento autónomo aleatorio, que requiere de sensores para detectar colisiones. Éstos producen un cambio de dirección aleatorio, de manera que el robot “rebota” ante los obstáculos. Es el método usado por la mayoría de los robots barredores automáticos nombrados anteriormente.

Ambos tipos de navegación requieren de la intervención humana en cierto grado: los robots sigue líneas pueden perder su referencia (por ejemplo después de sufrir un golpe o una colisión) y los robots de movimiento aleatorio corren el riesgo de quedarse atrapados bajo ciertas restricciones como una zona de difícil acceso, entre otros posibles problemas. En estas situaciones se hace imprescindible la actuación humana para devolverlos a su operación normal. Pero también existen robots capaces de recuperarse de estos problemas gracias a sistemas de navegación más complejos. Gracias a un conocimiento detallado del entorno (a través de un sistema de visión por ejemplo) y de la localización del robot (dada por este mismo sistema o por otro independiente), estos robots calculan su ruta al punto objetivo o de destino y la ejecutan de forma totalmente autónoma, y sin requerir de ninguna guía o camino previsto, ni ningún tipo de intervención humana. Además, algunos de estos robots pueden cambiar su trayectoria de forma dinámica, durante la ejecución del movimiento, al aparecer un obstáculo inesperado en la trayectoria. Ciertos robots que se encargan de trasladar material de un punto a otro en una fábrica cuentan con esta tecnología de navegación.

Teleoperación

Un robot teleoperado es aquel se encuentra total o parcialmente bajo el control de un sistema de manejo remoto, tras el que se encuentra un operador. Este dispositivo puede estar conectado directamente al robot o ser inalámbrico. Es habitual que este dispositivo también reciba información del robot, como las imágenes recogidas por una cámara, de manera que el operador pueda ver lo que está haciendo en tiempo real. Esto es útil, por ejemplo, cuando la distancia entre uno y otro es considerable, o si algún obstáculo dificulta la visibilidad del robot desde el punto de operación.

Normalmente la teleoperación se usa para mantener al operador lejos de posibles daños; ejemplo de esto son los robots desactivadores de explosivos a distancia, robots de patrulla o los ya mencionados robots reparadores espaciales. También se utilizan para ejercer distintas funciones de forma remota, aunque éstas no reviertan peligro, como los robots de cirugía asistida.

A parte de imágenes, el operador puede recibir más datos del robot como por ejemplo información tridimensional del espacio alrededor del robot o parámetros medidos por éste como presión o temperatura. En el caso de robots como los de cirugía asistida, el operador (en este caso el cirujano) recibe en el joystick de control una fuerza opuesta a la que ejerce él cuando el robot encuentra dificultad para realizar el movimiento que se le ordena, transmitiendo así una especie de sentido de tacto a quien lo maneja (Ostrovsky, 2009). A continuación puede verse un puesto de teleoperación de un robot de cirugía asistida que cuenta con esta tecnología:

IMAGEN 11 - CIRUJANO MANEJANDO UN ROBOT DE CIRUGÍA ASISTIDA TELEOPERADO

1.4. OBJETIVOS

Actualmente el manejo del robot Manfred se realiza a través de una interfaz que se ejecuta de manera local en el sistema operativo del robot, siendo necesaria por tanto una conexión física a éste (es decir, conectarle al menos un monitor y un teclado o ratón) o bien un acceso por escritorio remoto a través de los diferentes protocolos que existen para ello (como VNC, *Virtual Network Computing*). Si bien el robot podría operarse a distancia a través de este método no se considera que posea capacidad de teleoperación, ya que se trata de una pasarela de acceso al modo de operación local, el cual no está pensado ni optimizado para este uso.

El objetivo principal de este proyecto es, por tanto, dotar al robot manipulador Manfred de una interfaz de teleoperación real que permita controlar las distintas funciones del robot a distancia, y que ésta sea fácilmente accesible desde distintos tipos de dispositivo. Será necesario comenzar investigando qué tecnologías son las más idóneas tanto para la transmisión de datos entre el dispositivo de control y el robot como para el diseño de la interfaz, teniendo en cuenta que Manfred ya cuenta con un

sistema operativo sobre el que corre un software de control (Álvarez, 2011) al que la interfaz se tendrá que adaptar.

A continuación se desglosan y detallan las características que se pretenden conseguir en la interfaz que se desarrolla en este proyecto fin de carrera:

1. El modo de teleoperación del que se dotará al robot ha de ser inalámbrico, por lo que habrá que configurar una interfaz de este tipo en el equipo. Será por tanto necesario estudiar las distintas tecnologías de comunicación sin cables que existen actualmente y seleccionar la más idónea para este caso.
2. La interfaz debe funcionar o ser accesible desde distintos sistemas operativos y tipos de dispositivo. Se pretende que sea funcional independientemente del sistema operativo ejecutado en el dispositivo desde el que se acceda. Es deseable la compatibilidad con el máximo número posible de éstos.
3. El software diseñado debe funcionar en el sistema operativo del robot y utilizar el API (*Application Program Interface*) que proporciona el software de control ya existente en el equipo.

Capítulo 2. TECNOLOGÍAS ACTUALES

Este capítulo recoge el estado actual de las tecnologías que serán utilizadas en la realización de este proyecto, analizando sus ventajas e inconvenientes generales y particulares de su uso en la interfaz.

2.1. COMUNICACIONES INALÁMBRICAS

En la última década, las comunicaciones inalámbricas han evolucionado mucho tanto en capacidad de transmisión de datos y fiabilidad como en sencillez de uso. Esto ha provocado que su uso se extienda masivamente en la electrónica de consumo, lo que ha hecho que los precios de estos dispositivos hayan bajado hasta niveles muy asequibles.

La interfaz que se pretende desarrollar en este proyecto necesita ser accedida desde distancias relativamente cortas. Principalmente por esta razón las dos tecnologías que se estudiarán serán IEEE 802.11 (Wi-Fi) y Bluetooth.

2.1.1. IEEE 802.11 (WI-FI)

Se trata de un estándar de conexión para redes locales inalámbricas que se fijó por primera vez en abril del año 2000. En realidad Wi-Fi es el nombre comercial que adoptó la asociación de empresas (Wi-Fi Alliance) que certifica a los dispositivos compatibles con la norma estandarizada. El nombre del protocolo que estandariza esta norma es el IEEE 802.11 (la primera versión que se adoptó fue la IEEE 802.11b).

Actualmente, la mayor parte de los dispositivos portátiles de electrónica de consumo (ordenadores portátiles y PDAs o teléfonos inteligentes) cuentan con conexión Wi-Fi, lo cual es muy conveniente siendo un objetivo que la interfaz sea multiplataforma y fácilmente accesible desde diferentes tipos de dispositivo.

IMAGEN 12 - LOGOTIPO DE CERTIFICACIÓN WI-FI

Las conexiones Wi-Fi operan en la banda de 2,4 Ghz, disponible casi universalmente. Comenzó ofreciendo velocidades de transmisión de 11Mb/s, elevándose este número hasta los 54 Mb/s o 300 Mb/s en versiones posteriores del protocolo.

La tecnología Wi-Fi se ha expandido rápidamente gracias a sus múltiples e importantes ventajas:

- Comodidad: El hecho de eliminar cables de comunicaciones supone un ahorro (especialmente importante en el mundo empresarial, ya que reduce considerablemente la infraestructura necesaria y permite ampliaciones sin apenas inversión en nuevos equipos) y brinda la posibilidad de acceder cómodamente a la red desde cualquier lugar en su radio de cobertura (que es de unos 20 metros en interiores).
- Sencillez: Su uso, una vez configuradas estas redes, es muy sencillo incluso para usuarios no demasiado familiarizados con la informática.
- Compatibilidad: El sello de certificación Wi-Fi (ver Imagen 12) asegura la compatibilidad entre dispositivos de diferentes marcas que lo posean. La estandarización del protocolo colaboró en que los dispositivos sean asequibles, ya que facilitó la entrada de cualquier empresa a este mercado.

Como ya se ha mencionado antes, el uso de una tecnología tan sencilla y presente en la mayoría de los nuevos dispositivos electrónicos permite cumplir con comodidad con el objetivo de que la interfaz sea fácilmente accesible con independencia del tipo de dispositivo. Aunque el protocolo tiene algunas desventajas, asociadas a los riesgos de utilizar el aire como medio de transmisión:

- Velocidad: La velocidad de transmisión de datos es inferior a una conexión cableada.

- Seguridad: Es la desventaja principal. Existen programas y tarjetas capaces de capturar paquetes de conexiones ajenas y decodificar las claves de estas con la información contenida en ellos. Diversos estándares de seguridad (creados también por la Wi-Fi Alliance) han aumentado la robustez e inviolabilidad de estas redes, pero ninguno ha resultado ser infalible. Esto añadido a que no se puede controlar el área de cobertura de una conexión Wi-Fi, hacen a éstas poco recomendables para usos en los que la seguridad de la comunicación es crítica.
- Alcance: La potencia de una conexión Wi-Fi se puede ver afectada por interferencias de otras tecnologías de comunicación inalámbrica que operan en su misma frecuencia (como por ejemplo Bluetooth, GPRS, UMTS, etc.)

Estos inconvenientes no suponen una gran dificultad para alcanzar las cualidades deseadas para la interfaz: en el caso de la velocidad, aunque bien podría disminuir la responsividad de los controles, a la distancia a la que se pretende operar la transmisión de datos es suficientemente rápida. También el alcance de este tipo de redes es suficiente por esta razón. La seguridad de la conexión tampoco resulta una desventaja determinante ya que la información que se va a transmitir no es de carácter sensible, y la protección que ofrecen los protocolos de seguridad actuales será suficiente.

Dada la idoneidad de este tipo de conexiones para el uso que indican los objetivos del proyecto, y considerando que los inconvenientes no dificultan en exceso la consecución de los mismos, esta tecnología resulta bastante apropiada para su uso en el desarrollo de la interfaz.

2.1.2. BLUETOOTH

Bluetooth fue creado en 1994 por la compañía sueca de telecomunicaciones Ericsson como alternativa inalámbrica a los cables RS-232, pero debido a la expansión que comenzaban a experimentar los dispositivos portátiles, evolucionó rápidamente hasta convertirse en un protocolo de redes PAN (*Personal Area Network*, red de área personal) que permite intercambiar datos fácilmente entre varios dispositivos sin problemas de sincronización.

Aunque en su momento fue estandarizado como IEEE 802.15.1, actualmente ya no se mantiene, y es la asociación Bluetooth SIG (*Special Interest Group*) la que se encarga de desarrollar su especificación, de los procesos de certificación y de proteger a las marcas. Se trata de una asociación que agrupa a más de 20.000 empresas

tecnológicas. Para que un dispositivo pueda llevar el sello Bluetooth (ver Imagen 13) ha de ser certificado por la Bluetooth SIG. La tecnología Bluetooth está protegida por una serie de patentes que permiten su uso sólo en dispositivos certificados.

IMAGEN 13 - LOGOTIPO DE CERTIFICACIÓN BLUETOOTH

Al igual que las conexiones Wi-Fi, Bluetooth utiliza la banda de los 2.4 GHz, ofreciendo en su modo más básico tasas de transmisión de 1Mb/s. En versiones posteriores se introdujeron nuevos métodos de modulación que permitieron elevar esta cifra hasta los 24Mb/s, siempre que los dos dispositivos que se comunican sean compatibles.

A bajo nivel se trata de un protocolo del tipo maestro/esclavo, pero el rol que juega cada equipo puede cambiar de mutuo acuerdo si la funcionalidad así lo requiere, con lo que en realidad este comportamiento no es apreciable en protocolos que se implementen a más alto nivel. Cuenta con perfiles de funcionalidad, que descubren rápidamente las capacidades que poseen los equipos en comunicación y ahorran tiempo en el establecimiento de la comunicación. Algunos ejemplos son distribución de audio, impresión de imágenes, transmisión de archivos, manos libres, acceso a red, etcétera.

La seguridad en las conexiones está garantizada gracias al proceso de emparejamiento de dispositivos. Bien por demanda del usuario (por ejemplo al añadir un nuevo dispositivo) o bien al solicitar un servicio por primera vez, este proceso confirma la identidad de los dispositivos solicitando la intervención del usuario, por ejemplo, escribiendo en la pantalla del equipo solicitante un código que habrá que introducir en el solicitado. Este proceso no tiene por qué estar activo para todos los servicios, pudiendo ser alguno de éstos abierto a cualquier conexión.

El análisis de ventajas e inconvenientes del uso de Bluetooth para la interfaz que se pretende desarrollar resultaría muy similar al de Wi-Fi, ya que como se ha visto son tecnologías similares que comparten funcionalidades como establecimiento de redes, impresión remota, transmisión de archivos, etcétera. Pero lo cierto es que existen diferencias cuyo análisis puede resultar más interesante a la hora de seleccionar una u otra:

- Bluetooth es una tecnología que se centra en conexiones entre dispositivos portátiles y sus aplicaciones, mientras que Wi-Fi se centra más en reemplazar una red informática de área local, no sólo entre dispositivos móviles.
- La transmisión de datos en las conexiones Bluetooth es simétrica entre los participantes de la comunicación, al contrario que Wi-Fi cuya arquitectura es centrada en un punto de acceso por el que pasa todo el tráfico.
- Los procesos de autenticación también son diferentes: el emparejamiento de Bluetooth está preparado para admitir métodos muy simples, como pulsar un solo botón en una ventana de tiempo (en el caso de los auriculares manos libres, por ejemplo). Wi-Fi siempre requerirá una configuración mínima por parte del usuario, a menos que la seguridad esté desactivada.

2.2. DISPOSITIVOS PORTÁTILES

Los dispositivos electrónicos móviles (tablets, teléfonos inteligentes y ordenadores portátiles, principalmente) han sufrido una impresionante expansión en los últimos años. El desarrollo de nuevas tecnologías ha permitido que estos equipos sean cada vez más pequeños y asequibles, y les ha dotado de mayor autonomía, lo que los ha hecho mucho más apetecibles para el consumidor medio y provocado en gran medida dicha expansión. Actualmente la mayoría de las personas en España cuenta con un teléfono inteligente que usa a diario. Casi cualquier estudiante universitario tiene un ordenador portátil y, aunque aún no tan extendido, el uso de las tablets tiene una tendencia ascendente que se confirma año tras año (Fundación Telefónica, 2014). A continuación se encuentra un breve resumen de esta rápida evolución que permitirá tener una idea más precisa del estado actual de implantación y desarrollo de estos dispositivos:

Desde 1981 que se comenzó a popularizar el uso de los ordenadores portátiles (Velasco, 2011) éstos dispositivos han cambiado tanto que ni el aspecto exterior es remotamente parecido. El precio de estos equipos ha bajado de los 1.800 dólares que costaba el Osborne I (el primer ordenador portátil que tuvo cierto éxito comercial) hasta los 200 dólares por los que se puede conseguir actualmente un portátil de bajo coste “Chromebook” diseñado por Google en la tienda estadounidense de Amazon. Su peso es de un kilogramo y medio, mucho menos que los los 10,7 Kg que pesaba el Osborne I (sin baterías), con el aumento de movilidad que esto conlleva. Y sin entrar en comparar parámetros básicos de un ordenador (como cantidad y velocidad de memorias, velocidad de procesador, etcétera) el modelo actual de bajo coste cuenta

con tecnologías como Wi-Fi que en la época del portátil de Osborne Computer Corporation no exsistían. A parte de elevar de forma exponencial su penetración en los hogares, esta evolución ha permitido que se multipliquen los usos que tiene un ordenador portátil, convirtiéndolos en dispositivos cómodos, muy manejables, accesibles y versátiles.

Pero sin duda el desarrollo más impactante ha sido el de los teléfonos inteligentes y tablets. Se considera teléfono inteligente aquel que cuenta con un sistema operativo, y aunque existen desde 1992, cuando por primera vez IBM unió las funcionalidades de un teléfono con una PDA en un prototipo (Palazzesi, 2012), el primer ejemplo de lo que hoy comúnmente se conoce por teléfono inteligente (o por el término inglés *smartphone*, mucho más extendido) fue el iPhone de Apple, presentado y puesto a la venta en 2007. Su salida causó una gran expectación en todo el mundo, fue nombrado invento del año por la revista Time (Invention Of the Year: The iPhone, 2007). Sólo un año después fue lanzada una versión con mayor velocidad de transmisión de datos y disponible en más países que el primero.

También en 2007 fue lanzado Android, un nuevo sistema operativo para móviles creado por una nueva asociación de empresas de software, hardware y telecomunicaciones (la recién creada *Open Headset Alliance*) en colaboración con Google, que fue aún más determinante para la expansión que actualmente sufren los dispositivos móviles. Fue creado con intención de avanzar en estándares abiertos para los dispositivos móviles: proporciona un entorno de programación que permite que cualquier aplicación que se desarrolle con él funcione independientemente del dispositivo con Android en el que se ejecute. Esto ha causado que este sistema operativo alcance una cuota mundial de mercado de más del 75% en los teléfonos móviles (Fundación Telefónica, 2014), gracias a que los *smartphones* de las marcas Samsung, LG, HTC y Sony, entre otras muchas, lo ejecutan. El HTC Magic puede considerarse el primer teléfono equipado con este sistema operativo que alcanzó un nivel notable de ventas (Carmona, 2013), y con el que comenzó el desmesurado crecimiento de Android, y por tanto el de los *smartphones*, hasta los niveles actuales.

Unas de las ventajas decisivas para que Android se haya extendido tanto ha sido la variedad de plataformas sobre las que puede ejecutarse: tiene versiones para, entre otros dispositivos, relojes inteligentes, televisores, automóviles y *tablets* (ordenadores portátiles de mayor tamaño que un *smartphone* integrados en una pantalla táctil que se utiliza principalmente con el dedo o con un puntero especial). Android ha supuesto un enorme impulso en el desarrollo y las ventas de todos estos dispositivos, o de los modelos que cuentan con él. Y especialmente notable ha sido el caso de las *tablets*, que en la actualidad están experimentando su época de auge: por

ejemplo, se prevé que en 2015 las ventas de *tablets* superen a las de ordenadores portátiles a nivel mundial.

En resumen, a causa de distintos factores, en los últimos años los dispositivos portátiles se han convertido en aparatos cotidianos a los que se puede acceder fácilmente (debido a su bajo precio y/o a su extensión), y se espera que esta tendencia continúe en el futuro. Este factor los hace muy convenientes como soporte físico para la interfaz de teleoperación que se intenta desarrollar en este proyecto.

2.3. APLICACIONES WEB

Una aplicación web es una herramienta que se utiliza accediendo con un navegador a un servidor web a través de Internet o de una intranet. Se trata de aplicaciones codificadas en un lenguaje soportado por los navegadores web, y en las que se confía la ejecución a éstos. Ejemplos comunes de aplicación web son los calendarios colaborativos, tiendas en línea, *wikis* (sitios web fácilmente editables para compartir información) o los *webmail* (versión web de cliente de correo electrónico), como el utilizado por la Universidad:

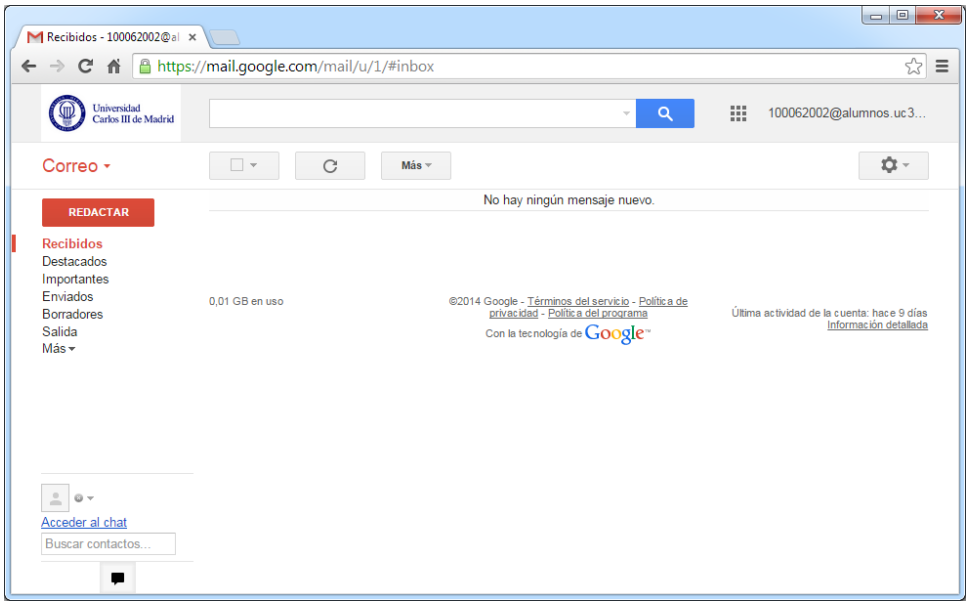


IMAGEN 14 - NAVEGADOR GOOGLE CHROME, EJECUTANDO EL *WEBMAIL* DE LA UC3M

En los comienzos de la computación cliente-servidor, cada aplicación tenía su propio software cliente que cada usuario tenía que instalar para utilizarla. Este programa hacía las funciones de interfaz, realizando peticiones al servidor y procesando las respuestas. Cualquier modificación o mejora en el servidor requería modificaciones en el software cliente, añadiendo dificultad y costes al uso de la aplicación.

Las aplicaciones web generan dinámicamente una serie de páginas en algún formato estándar, como HTML (*HyperText Markup Language*, lenguaje de marcas de hipertexto), que son interpretadas por el navegador del usuario. Estas páginas pueden contener fragmentos de código que son interpretados en el lado del cliente, permitiendo así añadir dinamismo a la interfaz. El lenguaje de este tipo más extendido es JavaScript.

El hecho de utilizar el navegador como cliente final tiene algunas desventajas. Funcionalidades comunes en aplicaciones de escritorio como dibujar en pantalla o arrastrar y soltar no son soportadas por las tecnologías web estándar. Debido al auge del uso de Internet, y el de este tipo de aplicaciones en particular, se han ido desarrollando tecnologías para implantar estas funcionalidades. Es destacable la técnica AJAX (*Asynchronous JavaScript And XML*, JavaScript y XML asíncronos) que permite mantener una comunicación asíncrona con el servidor en segundo plano, de forma que es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Otra desventaja de esta tecnología es que pueden surgir incompatibilidades con ciertos navegadores (o versiones antiguas de éstos), principalmente causadas por falta de adhesión de éstos con los estándares web. Además de esto, la posibilidad por parte del usuario de modificar características del dibujado de la página (como el tamaño y el color de la fuente) o inhabilitar ciertos elementos (como el intérprete de JavaScript) puede afectar a la estabilidad y al funcionamiento de la aplicación web. Tecnologías como Flash de Adobe o los *applets* de Java ignoran la configuración del navegador, pero la necesidad de plug-ins externos limita la compatibilidad, ya que los lenguajes utilizados no son estándares. Además, son más software que debe ser instalado y actualizado en el ordenador del usuario a parte del navegador.

Las aplicaciones web suelen estructurarse en tres capas, aunque este modelo no tiene porqué seguirse:

- El navegador web ofrece la primera capa. Es el encargado de la comunicación con el servidor, de dibujar la interfaz y de ejecutar las porciones de código encargadas del dinamismo de ésta.
- Un motor de web dinámica conforma la segunda capa. Se trata de un servicio que se ejecuta en el servidor, y se ocupa de generar las páginas web que serán enviadas al cliente y representadas en éste. Existen varias tecnologías que pueden utilizarse con este propósito como son PHP, Java Servlets, ASP o CGI.
- Por último, una base de datos constituye la tercera y última capa.

En resumen, las aplicaciones web cuentan con numerosas ventajas y escasos inconvenientes, que las hacen especialmente idóneas para el desarrollo de aplicaciones que necesitan comunicarse con un servidor:

- ✓ Ahorran tiempo, porque permiten realizar tareas sin descargar ni instalar ninguna aplicación.
- ✓ Con un navegador suficientemente actualizado no ofrecen problemas de compatibilidad.
- ✓ No ocupan espacio en el disco duro del usuario.
- ✓ Las actualizaciones son inmediatas, y sin ningún tipo de intervención por parte del usuario.
- ✓ A penas consumen recursos, ya que la mayoría de las tareas que realiza el software se realizan en el servidor.
- ✓ Son aplicaciones multiplataforma y portables, independientes del sistema operativo o equipo en el que se vayan a ejecutar, el único requisito es que cuenten con un navegador web (y conexión a Internet).
- ✓ La disponibilidad suele ser alta, ya que los servidores por norma son redundantes de forma que se asegura la continuidad del servicio.
- ✓ Al encontrarse los datos en el servidor, éstos son inmunes a ataques de virus.
- ✗ Habitualmente ofrecen menos funcionalidades que las aplicaciones de escritorio. Esto se debe a que las funcionalidades que se pueden realizar desde un navegador son más limitadas que las que se pueden realizar desde el sistema operativo.
- ✗ La disponibilidad depende de un tercero: el proveedor de la conexión a Internet o el que provee el enlace entre el servidor de la aplicación y el cliente. Así que la disponibilidad del servicio está supeditada a éste.

Dada la funcionalidad que requiere la interfaz que se plantea desarrollar, los inconvenientes que aquí se plantean no son demasiado determinantes, y teniendo presente que el principal objetivo de ésta es que sea multiplataforma y operable desde distintos dispositivos, una aplicación web parece la solución óptima.

Capítulo 3. PLATAFORMA EXPERIMENTAL (MANIPULADOR)

El manipulador móvil Manfred consta de una base de 2 grados de libertad y tipo diferencial y con un brazo de 6 g.d.l. y diseño antropomórfico. Esto resulta en un robot de 8 g.d.l. capaz de desenvolverse por entornos diseñados para humanos de una manera autónoma, siendo capaz de ocasionalmente abrir y atravesar puertas, evitar obstáculos o coger y manipular objetos (Blanco, Ansari, Castejón, López Boada, & Moreno, 2005).

A continuación se incluye una descripción más exhaustiva del manipulador, detallando cada uno de los sistemas, tanto de hardware como de software, que lo componen y permiten que funcionen con seguridad y autonomía.

3.1. HARDWARE

Para diseñar una interfaz funcional y práctica es necesario estudiar el hardware que se pretende manejar a través de esta.

3.1.1. ESTRUCTURA DEL ROBOT

El bastidor que sujeta e integra todos los elementos del manipulador Manfred tiene tres partes principales:

- Base móvil
- Torso fijo
- Brazo manipulador ligero

Base móvil: está construida en acero y unas dimensiones de 61 cm de diámetro y 65 cm de alto. Incluye las ruedas que permiten el movimiento del robot y las baterías necesarias para que lo haga de manera autónoma.

Torso fijo: es el bastidor principal del robot. Esta montado sobre la base y aloja la CPU (que lleva instalada la tarjeta PMAC2-PCI que permite el control de los motores) además de los cables de conexión de esta con el brazo, los cables de transmisión de potencia (que la llevan de la base a los motores del brazo), los servoamplificadores asociados a estos motores y el cableado de los sensores externos. Sobre este bastidor también se apoya la última parte, el brazo ligero, y dos sensores: el sensor láser y las cámaras del sistema de visión.

Brazo manipulador: está compuesto por elementos rígidos conectados por 6 articulaciones que le permiten alcanzar sendos grados de libertad. Combinando éstos este brazo dota a Manfred de la flexibilidad necesaria para acometer las tareas de agarre y desplazamiento de objetos. Es el elemento fundamental del robot.

En la siguiente imagen pueden apreciarse las tres partes descritas en las que se divide el robot:

IMAGEN 15 - PERFIL LATERAL DEL ROBOT MANFRED

3.1.2. SISTEMA SENSORIAL

Consiste en el sistema que permite al robot percibir el entorno donde se encuentra y con el que tiene que interactuar. Consiste en unos sensores que transforman las variables físicas que caracterizan dicho entorno en datos válidos para los distintos módulos del sistema de procesamiento (localización, sistema de seguridad, planificación de movimientos...). Este sistema consta de cuatro subsistemas, y cada uno de ellos cuenta con distintos sensores:

- Subsistema de telemetría láser
- Subsistema de visión
- Sensor fuerza/par
- Sensores cinemáticos

Subsistema de telemetría láser: este subsistema se encarga de modelar el espacio de trabajo que rodea al manipulador móvil. De esta manera el robot obtiene la información necesaria sobre las posiciones y distancias a los objetos que hay en su entorno, que será utilizada en las tareas de navegación y localización del robot durante su trabajo en un espacio amplio. Cuenta con dos sensores, uno 2D y otro 3D que son utilizados en función de la complejidad y el grado de ocupación de dicho espacio de trabajo:

- Telémetro láser Hokuyo UTM-30LX: colocado en la parte superior de la base del robot, cuenta con 270º de apertura y una distancia de detección desde 10 cm a 30 m. Tiene una frecuencia de escaneo de 40 Hz con una resolución de 0.25º, y se comunica con el ordenador interior a través de una interfaz USB 2.0. Consume 700mA a una tensión de 12V, lo que lo hace apropiado para alimentarlo con baterías.
- Telémetro láser bidimensional PLS de Sick: este sensor cuenta con 180º de apertura que permiten la adquisición de datos en 2D. Para conseguir datos en tres dimensiones está montado sobre un bastidor motorizado que permite un movimiento vertical de 45º. Para la navegación en entornos poco ocupados se utiliza su modo de funcionamiento en 2D: el escáner adquiere datos en un plano paralelo al suelo, en un entorno más complejo el sensor funciona en el modo 3D. El telémetro captura 361 medidas en un barrido horizontal (de 180º, cada 0.5º). El error máximo en la medida de profundidad es de 20 mm, y depende de dicha profundidad y del grado de apertura. Otras características del sensor son:
    - Rango máximo: 80 m
    - Resolución angular: 0.25/0.5/1º (configurable)

- Tiempo de respuesta: 26 ms
- Resolución de la medida: 10 mm
- Tasa de transferencia: 500 kbaud
- Alimentación: 24 V / 6 A

Subsistema de visión: el robot debe ser capaz de percibir el entorno de los objetos a manipular en tres dimensiones para reconocer dicho objeto y determinar su orientación y posición relativa respecto al manipulador (para así determinar la orientación necesaria para su manipulación). Este subsistema consta de los siguientes elementos:

- Cámaras de color: dos cámaras (una en blanco y negro y otra a color) se encargan de reconocer los objetos para estimar su posición y orientación respecto al robot: La primera, Sony EVI-D100, se encuentra en la parte frontal del cuerpo del robot, y para cuando el brazo interfiera en el campo de visión de ésta, el robot incorpora la segunda, una minicámara Sony XC-ES50CE, sobre la muñeca del brazo.

IMAGEN 16 - CÁMARAS DE COLOR DE MANFRED

- Cámara 3D: el robot también cuenta con una cámara 3D de tecnología *time-of-flight* que obtiene una matriz de distancias con los objetos situados en su campo de visión. La información devuelta por esta cámara se cruza con las imágenes de la cámara a color para poder realizar una segmentación de objetos mucho más precisa, permitiendo una manipulación más fina de éstos.

IMAGEN 17 - CÁMARA 3D DE MANFRED

Sensor fuerza/par: este sensor dota al robot de un sentido que podría compararse con el humano del tacto. Está situado en el extremo del brazo, uniendo éste al elemento terminal (por ejemplo, pinza). Se trata de un sensor fuerza/par JR3 modelo 67M25A-U560 (ver Imagen 18) cuyas características principales son: capacidad máxima de carga de 11 Kg, 175 g de peso y una frecuencia máxima de medición de 8 KHz. Proporciona medidas de fuerza y par en 3 ejes que son utilizadas en el lazo de control de fuerza del manipulador. El sensor combina las señales producidas por un sistema de galgas extensiométricas y un sistema de adquisición mediante DSP (Digital Signal Processor) gracias al cual puede obtener medidas con un elevado ancho de banda y SNR. Este sistema se encuentra en una tarjeta en el bus PCI del ordenador compatible con los sistemas Windows y GNU/Linux.

IMAGEN 18 - SENSOR FUERZA/PAR JR3

Sensores cinemáticos: son los sensores encargados de obtener la posición de cada articulación, para que así se puedan calcular, en el sistema de procesamiento, las magnitudes derivadas de ésta: velocidad y aceleración. Se encuentran acoplados a los ejes motores obteniendo la posición de éstos en forma de cuentas de *encoder*, pero ésta se trata de una posición relativa. Para obtener posiciones absolutas el robot cuenta con una rutina de arranque (o función *homing*) que lo coloca en una posición inicial de manera que a partir de esta pueden calcularse posiciones absolutas.

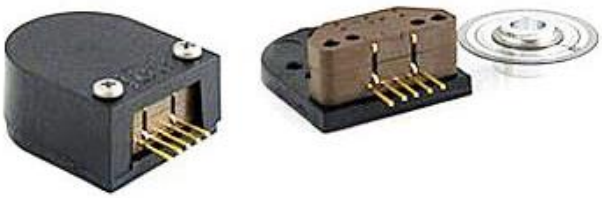


IMAGEN 19 - *ENCODERS* ÓPTICOS HP HEDS 550

Los sensores utilizados son *encoders* ópticos de alta resolución HP HEDS 550 (ver Imagen 19) cuyas características principales están detalladas en la siguiente tabla:

Resolución	1024 cuentas/revolución
Alimentación	5 V
Marca de índice	Sí

Para la función *homing* se utilizan sensores inductivos de 3 mm de diámetro con una distancia de detección de 1 mm.

3.1.3. Sistema locomotor

La base citada en el apartado 3.1.1 es la que constituye el sistema locomotor del robot. Cuenta con cinco ruedas: tres ruedas de apoyo que garantizan la estabilidad del robot y facilitan su movimiento, y dos ruedas motrices asociadas a dos motores *brushless* (sin escobillas) y a sus correspondiente servoamplificadores. Estas dos ruedas producen un movimiento diferencial que permiten al manipulador avanzar, retroceder y girar sobre sí mismo. Estas ecuaciones expresan la cinemática directa e inversa de su movimiento:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1/2 & -1/2 \\ 1/d & 1/d \end{bmatrix} \cdot \begin{bmatrix} v_d \\ v_i \end{bmatrix}$$

donde v y ω son respectivamente las velocidades lineal y angular de la base, v_d y v_i las velocidades de las ruedas derecha e izquierda y d la distancia entre los ejes de las ruedas. A continuación se muestra una imagen esquemática de la base en la que puede apreciarse la disposición de las ruedas:

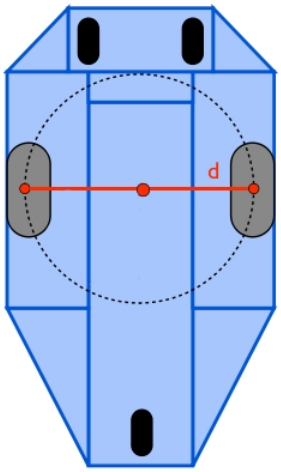


Imagen 20 - Esquema de la base de Manfred

3.1.4. Sistema de alimentación

También en la base se encuentra el sistema de alimentación que permite al robot operar con autonomía. Consiste en cuatro baterías de 12 V conectadas en serie

de manera que proporcionan una tensión de 48 V en corriente continua. Además, como sistema de seguridad, la tensión y la corriente proporcionadas por la batería son continuamente monitorizadas por un microcontrolador PIC16F818 que envía esta información al ordenador de control, y además, de manera autónoma, realiza una parada automática controlada de los motores en caso de que la tensión sea demasiado baja o que la corriente supere un nivel umbral. Las baterías son modelo LC-X1242P de Panasonic, y tienen una capacidad de 42 A/h.

3.1.5. Sistema manipulador LWR-UC3M-1

El encargado de realizar las tareas de manipulación es el brazo robótico de 6 grados de libertad que ya se ha mencionado en un apartado anterior. Se trata del brazo robótico ligero LWR-UC3M-1, el cual ha sido desarrollado íntegramente en la Universidad Carlos III de Madrid. Sus características principales son:

- Redundancia cinemática similar al brazo humano.
- Peso de 18 kg.
- Capacidad de carga máxima en el extremo del brazo de 4,5 kg.
- Relación carga/peso entre 1:3 y 1:4.
- Alcance en torno a 955 mm.

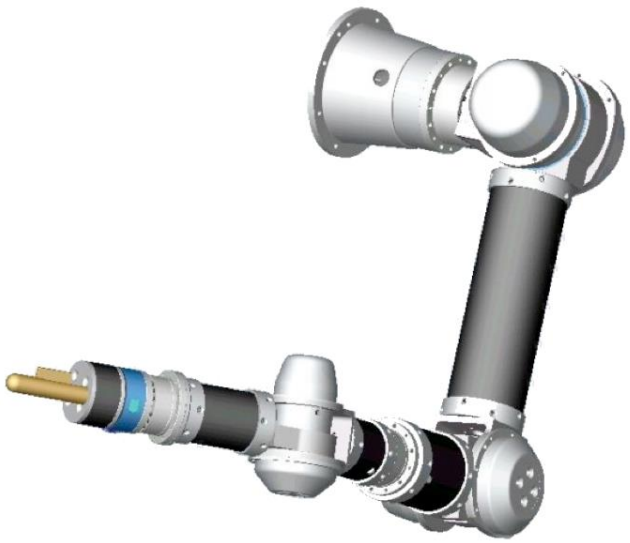


Imagen 21 - Diseño 3D del manipulador LWR-UC3M-1

Este brazo se monta lateralmente en el bastidor principal de forma que interfiera lo menos posible en el campo de visión de las distintas cámaras con las que cuenta el robot. La tracción para las articulaciones es proporcionada por motores

brushless de corriente continua conectados a una etapa de reducción de velocidad y aumento de par del tipo Harmonic Drive (concretamente Harmonic Drive AG HFUC-2UH) que incorpora rodamientos de salida integrados.

Función *homing*

Como ya se ha mencionado antes, los *encoders* que se encargan de controlar la posición del brazo son del tipo relativo, con lo que si se quiere conocer la posición absoluta de las articulaciones se debe partir de una posición inicial (o posición *home*). Para alcanzarla el robot tiene programada una función, en el lenguaje propio de la tarjeta de control PMAC, conocida como *homing*, que establece la posición inicial antes mencionada: aquella en la que todos los eslabones están estirados apuntando hacia el suelo (elegida porque para llegar a ella la mayoría de motores no tienen que hacer ningún trabajo, con el consiguiente ahorro de energía). Una vez alcanzada esta posición se almacenan en unas variables internas de la tarjeta los valores que tienen los enconders en este momento, con lo que ya pueden calcularse las posiciones absolutas.

3.1.6. Sistema de control. Tarjeta PMAC

Se trata de la tarjeta donde se procesa el control de los motores del robot: el modelo PMAC2-PCI (*Programable Multi-Axis Controller*, controlador multi-eje programable) de la compañía Delta Tau Data Systems, Inc. Esta tarjeta puede controlar hasta 8 ejes simultáneamente con un alto nivel de precisión (Álvarez, 2011). Puede ejecutar distintos tipos de movimiento que permiten hacer distintos tipos de control en la trayectoria a ejecutar.

Este modelo de tarjeta es altamente configurable y muy versátil, pero no está diseñada para ser conectada directamente a los drivers de potencia que controlan finalmente los motores, o a los *encoders* que informan de su posición. El accesorio modelo 8E de la misma marca cuenta con la lógica necesaria para interaccionar con 2 motores de Manfred por medio de cuatro conversores D/A (con lo que éste cuenta con 4 tarjetas de dicho modelo).

Entre los accesorios 8E y cada motor, Manfred cuenta con 8 drivers modelo BE25A20 de Advanced Motion Controls, que se encargan de general la señal apropiada para cada motor en función del nivel correspondiente de tensión que reciben de los adaptadores 8E.

3.2. SOFTWARE

3.2.1. SISTEMA OPERATIVO

La CPU que controla a Manfred funciona con un sistema operativo de la familia GNU/Linux, en concreto, la distribución Ubuntu, patrocinada por Canonical Ltd. en su versión 11.10, alias *Oneiric Ocelot* (Ocelote Onírico).

Este sistema operativo, como todos del tipo GNU/Linux, cuenta con varias ventajas:

- Está basados en un núcleo muy estable y robusto: el núcleo Linux.
- Es multitarea, multiplataforma y multiusuario.
- Es gratuito. A pesar de esto, existe la posibilidad de obtener soporte profesional (y de pago) a través de Canonical.
- Estos sistemas operativos cuentan con carga selectiva de programas según la necesidad, lo que los hace muy convenientes para equipos embebidos.

Además de estas ventajas generales, Ubuntu ofrece una gran facilidad para instalar el software necesario para la interfaz (un servidor web y módulos adicionales de ROS). Además, los sistemas operativos GNU/Linux son los más utilizados en servidores web a nivel mundial (Usage Statistics and Market Share of Operating Systems for Websites, January 2015), lo que asegura una amplia documentación, tutoriales de todo tipo y un buen soporte comunitario.

3.2.2. ROS – ROBOT OPERATING SYSTEM

Se trata de un entorno de programación (*framework*, en inglés) para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo al programador: abstracción del nivel hardware, control a bajo nivel de dispositivos y comunicación entre procesos entre otros. Se desarrolló en 2007 por el Laboratorio de Inteligencia Artificial de Stanford para su proyecto STAIR, pero actualmente su desarrollo continúa casi en exclusiva en el instituto Willow Garage de California, donde colaboran más de veinte instituciones. Está escrito en C++ y Python.

En el sistema operativo del robot Manfred está instalada la versión *Electric Emys* de ROS.

A continuación se enumeran las características más importantes de ROS:

- Es muy ligero, y facilita que las aplicaciones que se desarrollan con él lo sean.

- El código que se escribe en ROS puede ser utilizado con pocos cambios en otras plataformas similares.
- Permite varios lenguajes de programación. Ya existen nodos implementados en Python, C++ o Lisp, y existen librerías experimentales en Java y Lua.
- Cuenta con una extensión orientada a pruebas que facilita la depuración.
- Es apropiado para el funcionamiento de múltiples nodos.

A nivel lógico, estos son los conceptos lógicos básicos de la red de procesos que implementa ROS:

- *Node* (nodo): unidad básica de procesamiento de ROS. Utilizan las librerías cliente que proporciona ROS, por ejemplo la librería *roscpp* (si el nodo está codificado en C++). Son el equivalente a un ejecutable en otro sistema operativo.
- *Master node* (Nodo maestro): se trata del nodo principal de ROS. Proporciona servicios de registro y búsqueda de nombres, que permiten que los *nodes* puedan encontrarse y les dota de mecanismos para comunicarse: *topics* y *services*, a parte de otras funciones.
- *Parameter server*: permite almacenar datos de manera global, forma parte del nodo *master*.
- *Bags*: almacenes de datos.
- *Message*: unidad de datos que se intercambian los procesos.
- *Topic*: flujo de datos de forma publicador/suscriptor. El nodo publicador registra el *topic* en el nodo *master* y luego le envía datos. El *master* los encamina a los nodos suscriptores. En el mismo *topic* puede haber varios publicadores o suscriptores sin que ninguno tenga consciencia de la existencia de los demás, la información se comparte de forma desacoplada.
- *Service* (Servicio): modelo de comunicación cliente/servidor. Un nodo ofrece un *service* (a través del nodo *master*) bajo un nombre concreto al que el resto de los nodos pueden enviar peticiones (que pueden contener datos). El nodo debe ocuparse de responder estas peticiones de forma conveniente: la respuesta puede ser un mensaje con datos o una acción sobre el sistema que no involucre comunicación.

El sistema de archivos de ROS se organiza con los siguientes elementos:

- *Packages* (Paquetes): constituyen la unidad principal de organización de software. Pueden contener nodos, librerías o archivos de configuración.
- Archivos Manifest: contienen información sobre un paquete.
- *Stacks* (Pilas): conjunto de paquetes relacionados por una funcionalidad específica.
- *Message types* (tipos de mensaje): archivos donde se definen las estructuras de los mensajes que usan los *nodes* para comunicarse entre sí.
- *Services* (Servicios): archivos donde se definen las estructuras de las peticiones y las respuestas de un *service*.

Actualmente ROS sólo funciona sobre plataformas basadas en UNIX. Está ampliamente probado sobre Ubuntu, que es el único sistema operativo que soporta de forma oficial, aunque la comunidad de usuarios ofrece soporte para otras distribuciones como Debian, Fedora, Gentoo y Arch Linux, entre otras.

3.2.3. Sistema de procesamiento

Se trata del software que interactúa con la tarjeta de control PMAC y ejecuta las tareas correspondientes al control coordinado de todo el robot. Este software también ha sido íntegramente desarrollado en la Universidad.

Proporciona una API que permite a otros módulos o nodos del sistema operar con el robot, haciéndoles transparente las tareas de control. Los comandos simples que provee esta interfaz permiten, por ejemplo, mover una articulación determinada un número de grados o ejecutar una trayectoria compuesta que implique a varias articulaciones moviéndose simultáneamente. También permite introducir de forma sencilla correcciones en una trayectoria durante su ejecución, algo frecuente en los entornos de trabajo dinámicos para los que Manfred ha sido diseñado (Álvarez, 2011).

Dada la complejidad del sistema y a que ROS ofrece (y facilita, como ya se ha visto) esta posibilidad, se optó por el desarrollo de módulos independientes que trabajan conjuntamente por medio del intercambio de datos y que son individualmente sensibles a fallos. Esto proporciona un software fiable pero también escalable, algo muy importante en sistemas complejos.

Capítulo 4. SOLUCIÓN PROPUESTA

4.1. RESUMEN

Teniendo en cuenta todo lo desarrollado sobre el estado actual de las tecnologías que son susceptibles de ser utilizadas (ver Capítulo 2) y las ventajas y/o limitaciones que marca el hardware y software actuales del robot, la solución propuesta será una aplicación web dinámica alojada en un servidor web Apache instalado en el sistema operativo ya existente en el robot. Esta aplicación web intercambiará información con el sistema de procesamiento (ver 3.2.3) a través de *rosbridge*, un paquete de ROS que ofrece compatibilidad con los mecanismos de comunicación internos de este *framework* (ver 3.2.2) a aplicaciones no desarrolladas bajo él.

Como se indica en el apartado 2.3, en el desarrollo de una aplicación web intervienen distintos elementos y lenguajes de programación. La combinación de éstos permite crear aplicaciones dinámicas e interactivas con relativa sencillez.

A continuación se detalla cada uno de los elementos nuevos que componen y hacen funcionar a la interfaz.

4.2. SERVIDOR HTTP: APACHE

Como ya se ha mencionado, el servidor HTTP elegido para alojar la web de control remoto de Manfred ha sido Apache. Se trata del servidor más utilizado desde 1996 en todo el mundo (Netcraft, 2014), aunque en los últimos años su uso haya tendido levemente a la baja. Este servidor jugó un papel clave en el crecimiento inicial de la WWW debido a que es de código abierto y gratuito, lo que propició su rápida extensión. Además es modular, extensible, y muy estable, consecuencia de todos estos años de uso masivo y desarrollo comunitario.

Por la misma razón cuenta con un muy buen soporte comunitario ante cualquier nuevo problema o agujero de seguridad que pueda surgir. El hecho de que sea de código abierto hace la corrección de errores y/o vulnerabilidades sea muy rápida, y además el sistema operativo se encarga de aplicarlas automáticamente, con lo que elegir esta combinación (Ubuntu + Apache) garantiza la seguridad. Además, el hecho de que Apache sea modular facilita la instalación y el uso de motores de web dinámicas (en este caso, PHP).

Para instalar Apache en Ubuntu simplemente hay que elegirlo en cualquiera de sus gestores de paquetes (*apt-get* en la línea de comandos, o *Synaptic* en el escritorio, por ejemplo), aunque existe una forma más sencilla de instalarlo junto al motor PHP que se verá más adelante. Tras su instalación (y reiniciar el sistema) el servidor comenzará a servir las páginas que se encuentran en la carpeta /var/www del sistema operativo. Inicialmente sólo se encuentra un archivo HTML sencillo con el texto "*It works!*" (¡Funciona!). Esta página puede verse escribiendo *http://localhost* en un navegador en el sistema, o escribiendo la IP del robot desde otro ordenador conectado a la misma red.

4.3. Motor de web dinámica: PHP

El dinamismo de las páginas que componen la interfaz se ha codificado utilizando el lenguaje de *scripting* PHP (*PHP Hipertext Pre-processor*, Pre-procesador de hipertexto PHP, se trata de un acrónimo recursivo). Este código es interpretado en el servidor y genera bajo demanda las páginas web que conforman la interfaz. La elección de este método se debe a varios factores:

- Es seguro y estable. Tras 19 años de historia ha sido largamente probado y perfeccionado por empresas y organizaciones de todo el mundo.
- Es sencillo. Posee una sintaxis limpia y tipado dinámico, aunque puede chequearse el tipo de las variables en tiempo de ejecución. Además, aunque no es obligatorio, el código PHP está integrado con el resto de código HTML como una etiqueta más, lo que facilita el diseño y la comprensión del conjunto.
- Como el resto del software descrito hasta ahora, es libre, de código abierto y gratuito.
- Posee una amplia documentación en su sitio web oficial. Además, al ser bastante popular, tiene un gran soporte proporcionado por la comunidad.

Instalar el intérprete de PHP en Ubuntu junto con Apache es tarea sencilla: sólo hay que instalar el paquete del módulo de PHP para Apache *libapache2-mod-php5* y el resto de paquetes necesarios se instalarán automáticamente, ya que éste depende de ellos.

Cuando el intérprete se encuentra corriendo junto al servidor, los archivos con extensión .php (esto es configurable, aunque el valor por defecto resulta apropiado) serán pre-procesados en busca de código PHP antes de que el servidor los envíe al navegador cuando este los pida. Si este código existe será interpretado y su resultado formará parte de las páginas enviadas.

En la interfaz, el pre-proceso de PHP se utiliza en distintas funcionalidades, como en el registro de entrada o el memorizado de trayectorias.

4.4. INTERFAZ

La parte visible de la interfaz ha sido desarrollada con los lenguajes más comunes en este tipo de aplicaciones: HTML, CSS (*Cascading Style Sheets*, hojas de estilo en cascada) y Javascript.

El diseño estático está escrito utilizando los lenguajes de descripción HTML y CSS. Por un lado HTML describe el contenido o forma básica de las páginas y los elementos que la van a componer, mientras que las hojas CSS se encargan de dar formato y estilo a estos elementos. Por ejemplo, en la interfaz de Manfred, el esqueleto de la página y las cajas de los diferentes controles están descritos en HTML mientras que el color de estos elementos o el grosor de sus bordes lo están en las hojas CSS.

Por último, el dinamismo visual de la interfaz y el intercambio de datos con el software de control del robot se realizará en Javascript. Se trata de otro lenguaje de programación interpretado, pero al contrario que PHP, en el navegador web cliente. Tiene una sintaxis similar a la de C, y cuenta con una implementación del DOM (*Document Object Model*, modelo del documento de objetos) que permite interactuar con los elementos la página web. La comunicación con los nodos de ROS, que ocurrirá en segundo plano, también estará escrita en Javascript, utilizando los recursos que proporciona la librería *roslibjs*, que será analizada en el apartado 4.5.

Librería jQuery

Para el desarrollo del código Javascript se ha utilizado la biblioteca jQuery. Esta biblioteca simplifica la forma de interactuar con el DOM permitiendo manejar eventos y añadiendo algunas funcionalidades y efectos. Es también software libre y de código

abierto, utilizado por multitud de empresas (como Nokia o Microsoft). En la interfaz también se utilizan algunos de los efectos visuales que esta librería ofrece. jQuery también implementa funciones AJAX (intercambio asíncrono de datos con el servidor) pero esta funcionalidad no se utiliza en la interfaz; como ya se ha explicado el intercambio de datos con ROS corre a cargo de la librería *roslibjs*.

4.5. Conexión con ROS: rosbridge – rosjs

El tándem formado por el paquete de ROS *rosbridge* y la librería Javascript *rosjs* permite la comunicación entre la aplicación web y los nodos que controlan el robot.

El paquete *rosbridge* ofrece una API que permite la comunicación de los nodos del sistema con elementos externos a ROS a través de, entre otras opciones, un servidor de WebSockets (tecnología que permite al navegador que la posee establecer una comunicación bidireccional sobre un socket TCP). Permite a elementos externos a ROS publicar o suscribirse a *topics* e invocar *services* recibiendo la respuesta. La información es intercambiada en formato JSON (*JavaScript Object Notation*, notación de objetos de Javascript), como la del siguiente ejemplo:

```
{"menu": {
   "id": "file",
   "value": "File",
   "popup": {
     "menuitem": [
       {"value": "New", "onclick": "CreateNewDoc()"},
       {"value": "Open", "onclick": "OpenDoc()"},
       {"value": "Close", "onclick": "CloseDoc()"}
     ]
   }
 }
}
```

Imagen 22 - Ejemplo de datos en formato JSON

Por su parte, la interfaz utiliza la librería *rosjs* para comunicarse con *rosbridge*. Esta librería proporciona funciones de Javascript para establecer y monitorizar la conexión con el servidor de Websockets; y las herramientas para suscribirse o publicar *topics*, o llamar a *services* y obtener su respuesta.

4.6. Funcionamiento de la interfaz

4.6.1. Control de acceso

La interfaz de Manfred cuenta con un control de acceso a los controles del robot. Para acceder, será necesario introducir un nombre de usuario y una contraseña previamente registrados en los ficheros de configuración de la interfaz (ver 4.11.1). Esta es una imagen de la pantalla de acceso de la aplicación:

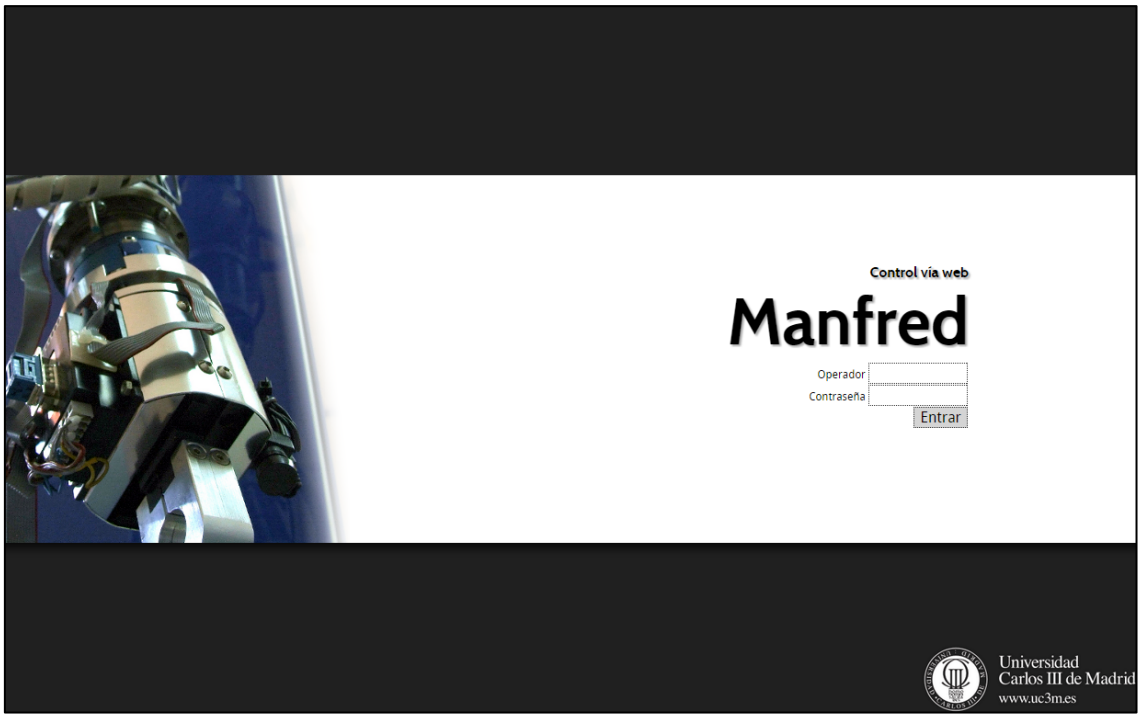


IMAGEN 23 - PANTALLA DE ACCESO A LA INTERFAZ

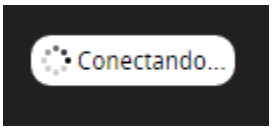
Al pulsar el botón "Entrar" la interfaz comprobará que el usuario existe y que la contraseña es correcta. Si esto es así mostrará la pantalla de control del robot, descrita en el siguiente apartado. Este registro tiene una validez de 30 minutos (configurable), pero la interfaz no deja de funcionar una vez transcurrido este tiempo; sólo será necesario volver a realizar el registro en caso de recargarla (por ejemplo por cerrarla o por pulsar el botón actualizar del navegador).

4.6.2. PANTALLA DE CONTROL

La pantalla de control del robot consta de seis secciones diferenciadas en cajas. Cada una de ellas ofrece distintos comandos y tipos de control del robot. La imagen 24 corresponde a una captura global de la interfaz abierta en un navegador de un ordenador.

A parte de las seis cajas pueden verse otros elementos:

- Estado de la conexión: aparece en la esquina superior izquierda de la web, e informa del estado de la conexión de la interfaz con ROS, o mejor dicho, con el nodo *rosbridge*. Puede mostrar distintas indicaciones:
    - Conexión en proceso: la comunicación por Websockets con el servidor de *rosbridge* está estableciéndose.



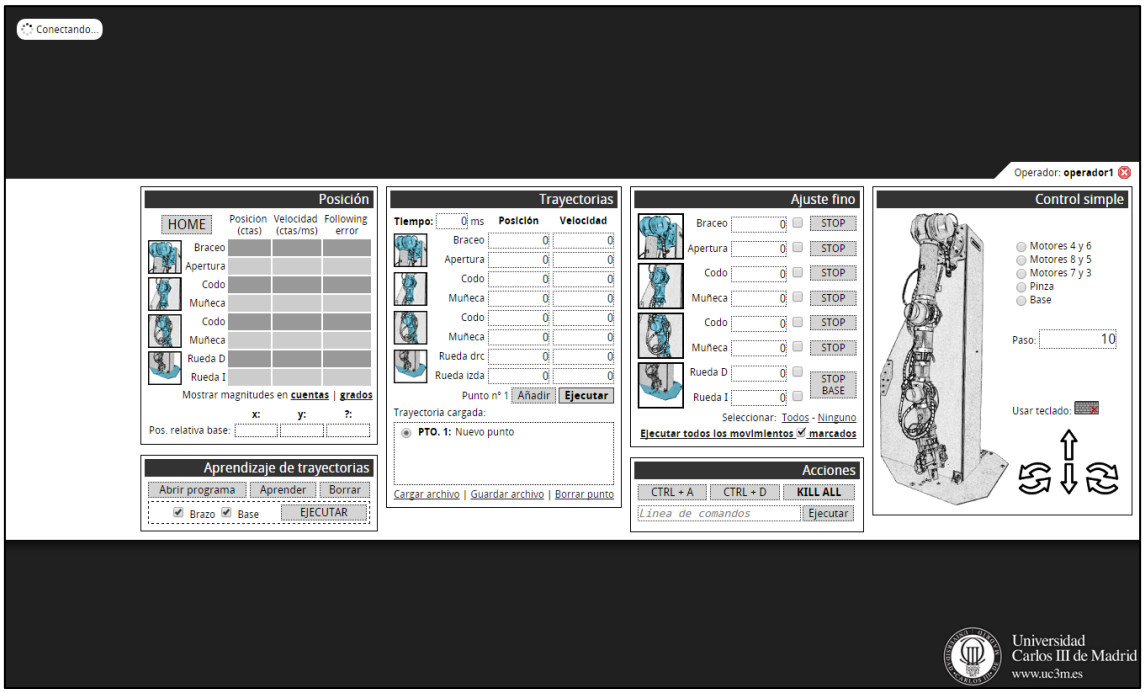
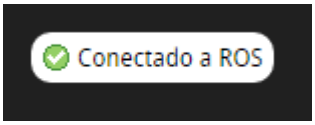
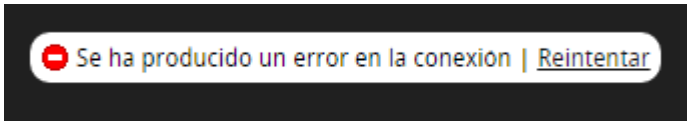


IMAGEN 24 - PANTALLA DE CONTROL DE LA INTERFAZ EN UN ORDENADOR

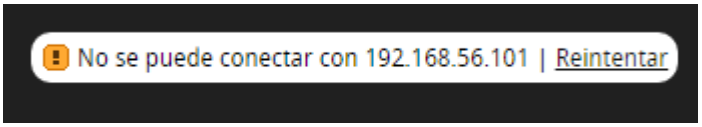
- Conexión correcta: la conexión se ha establecido de forma correcta y es posible utilizar la interfaz. Esta indicación desaparece al cabo de 5 segundos de mostrarse.



- Error en la conexión: si se produce algún error en el protocolo de comunicación entre *roslibjs* y el nodo *rosbridge* se mostrará esta indicación. Incluye un botón de reintentar para volver a iniciar la comunicación.



- No se puede establecer la conexión: la interfaz no encuentra el servidor de Websockets en la IP que tiene configurada. Hay varias posibilidades para que esto ocurra, como que el nodo *rosbridge* no esté en ejecución o que no haya una ruta de red hasta él. Dicha IP aparece en el mensaje para facilitar el diagnóstico de errores. También incluye un botón para reintentar la conexión.



- Pestaña de operador: se encuentra en la parte superior derecha de la banda blanca central de la página, donde se encuentran los controles, a

modo de pestaña. Muestra el usuario con el que se ha entrado al control y tiene un botón para cerrar la sesión de forma que sea necesario volver a ingresar con unas credenciales válidas para volver a utilizarla.

A continuación se desarrollará más la función y el uso de cada una de las partes principales de la interfaz de control:

4.6.2.1. CONTROL SIMPLE

Se trata de los controles más inmediatos para mover cualquier eje del robot o sus ruedas. Sólo hay que seleccionar la pareja de articulaciones que se desee mover (o la base) en el esquema interactivo haciendo clic sobre ella: al pasar el ratón por encima se iluminará en azul, color que permanecerá al pulsar el botón del ratón. Una vez seleccionada, cada pulsación de las flechas ejecutará un movimiento cuya amplitud dependerá de la cifra que se encuentre en el campo "Paso" del control en ese momento: éste se corresponde con el número de grados sexagesimales que gire la articulación correspondiente, excepto en el caso del avance o retroceso de la base, que será el número de centímetros de desplazamiento en línea recta.

Para que la correspondencia entre las direcciones que indican las flechas de control y el movimiento que hará el robot sea la esperada, el operador debe colocarse (física o figuradamente) detrás del robot. Una vez fijado el punto de referencia, las parejas de articulaciones en los que se ha dividido el robot para este control cuentan con una articulación radial, similar al codo humano, y otra axial, similar a la muñeca.

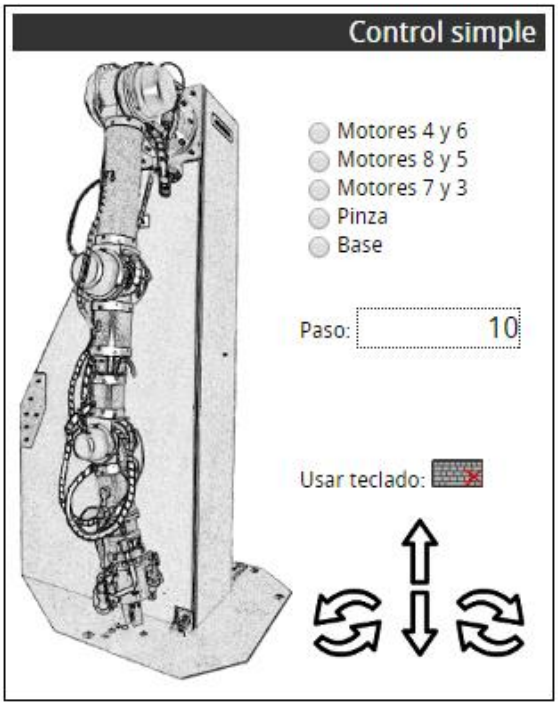


IMAGEN 25 - CONTROL SIMPLE DE LA INTERFAZ

Las flechas verticales ejecutarán el movimiento radial y las de giro u horizontales el axial. Por ejemplo, si se selecciona el grupo de articulaciones 4 y 6 (el "hombro" de Manfred) y se pulsa la flecha hacia arriba el brazo se elevará hacia delante:

En lugar de hacer clic sobre los botones de las flechas también es posible utilizar las teclas direccionales del teclado. Esta función inicialmente se encuentra inactiva para evitar movimientos accidentales, dado que estas teclas son muy utilizadas y es fácil que se pulsen con otro propósito, pero puede ser activada con el botón rotulado como "Usar teclado". El botón muestra el estado de la función apareciendo el teclado tachado cuando está desactivada.

A la derecha del esquema del robot aparecen unos selectores que cambian automáticamente al seleccionar una articulación en éste. En las etiquetas de estos selectores aparecen los números de motor que utiliza internamente el sistema de control de Manfred. Esta numeración se detallará en el capítulo 4.7.

4.6.2.2. AJUSTE FINO

Este control permite ejecutar movimientos compuestos por una o varias articulaciones de forma sencilla. Posee un campo por cada articulación similar al campo "Paso" del control simple, junto con una casilla de activación/desactivación de la articulación:

IMAGEN 26 - CONTROL "AJUSTE FINO"

Las articulaciones están ordenadas en orden descendente desde el hombro hasta la punta del brazo, una imagen descriptiva acompaña a cada pareja de campos. Tras completar las casillas correspondientes con los valores deseados y activar estas articulaciones, al pulsar el botón "Ejecutar todos los movimientos marcados" el robot comenzará a moverse a una velocidad uniforme, realizando todos los movimientos solicitados al mismo tiempo.

Si se desea detener alguna articulación antes de que el movimiento finalice, cada campo cuenta con un botón de parada individual que detendrá el movimiento de forma suave. Estos botones pueden utilizarse también junto con el resto de controles. Este cuadro también cuenta con dos atajos para marcar o desmarcar todas las articulaciones rápidamente.

4.6.2.3. ACCIONES

Está formado por tres botones que realizan distintas acciones y una línea de comandos con la que se puede interactuar directamente con la API del sistema de control de Manfred (ver 3.2.3):

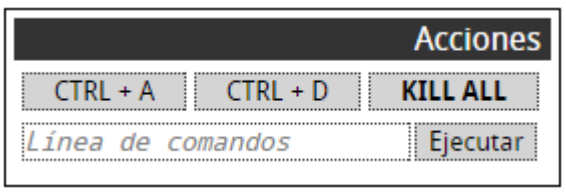


IMAGEN 27 - CUADRO DE CONTROL "ACCIONES"

Las funciones de los tres botones corresponden a comandos implementados en el software comercial de fábrica de la tarjeta PMAC. Las órdenes están nombradas según el atajo de teclado asociado en dicho programa:

- CTRL+A: abortar cualquier movimiento o programa.
- CTRL+D: deshabilitar todos los programas PLC y PLLC.
- CTRL+K ó KILL ALL: destruir todas las señales de alimentación de todos los motores.

Esta última orden resulta especialmente interesante. Al recibir este comando, el sistema de control desactiva la alimentación de todos los motores del robot de forma inmediata e incondicional. Este comando se hace imprescindible ya que cualquier movimiento que realiza Manfred conlleva el riesgo de dañar al propio robot o a algún elemento o persona cercanos. Su función es similar a la de una "seta" de parada de emergencia como la que poseen un gran número de equipos eléctricos de gran tamaño. Para obtener más información acerca de estas órdenes puede consultarse el manual de referencia software de la tarjeta PMAC (DELTA TAU Data Systems, 2004).

En la línea de comandos pueden escribirse comandos del tipo *online* para ejecutar movimientos u otras acciones en los motores. Los comandos introducidos son enviados directamente a la tarjeta PMAC sin ninguna condición o procesamiento previos. El envío puede realizarse, aparte de utilizando el botón "Ejecutar", pulsando la tecla Intro. Si el comando escrito genera una respuesta, ésta será presentada al usuario en un nuevo cuadro que aparecerá bajo la caja de texto donde se ha introducido la petición.

4.6.2.4. TRAYECTORIAS

Este cuadro permite ejecutar en Manfred movimientos complejos en los que la posición y la velocidad de cada articulación del robot están controladas en todo momento. Además es capaz de memorizar estas trayectorias, editar alguno de sus puntos, añadir puntos nuevos, descargarlas a un fichero o importarlas desde uno de éstos.

En la parte superior del cuadro se encuentran los campos donde se introducen los datos de cada punto definido de la trayectoria. Bajo ésta, se encuentra la lista de puntos ya cargados. Cada punto está definido por un valor de tiempo (relativo al inicio del movimiento) más una pareja de coordenadas posición/velocidad por cada articulación. En el control, el primer cuadro de texto corresponde al tiempo mientras

que las coordenadas se introducen en la tabla justo a continuación. Entre estas dos partes se encuentra el botón “Ejecutar” que envía al robot la trayectoria.

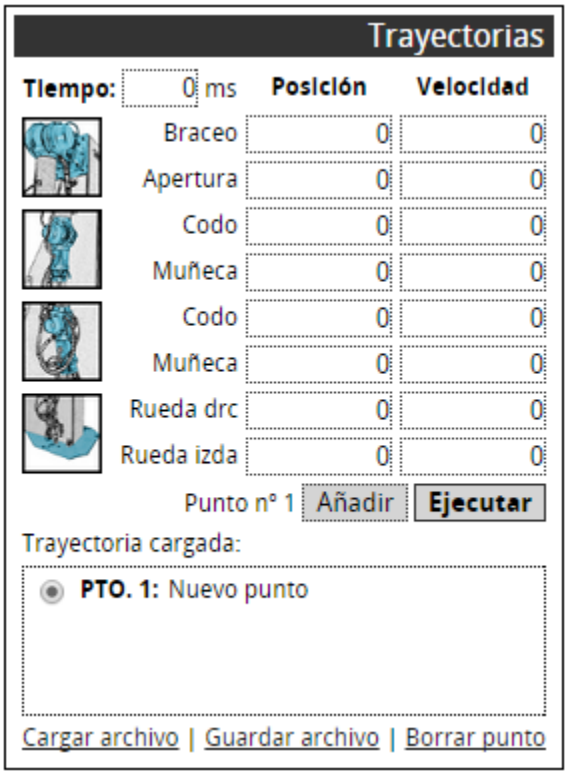


IMAGEN 28 - CUADRO DE TRAYECTORIAS

Como ya se ha mencionado, en el cuadro “Trayectoria cargada” aparecerán los puntos ya introducidos con un botón de selección asociado a cada uno de ellos. Por encima de éstos, como si fuese el primer punto, aparecerá la opción de añadir un nuevo punto. Para añadir un nuevo punto hay que seleccionar esta opción e introducir las coordenadas del mismo en la parte superior. Al pulsar el botón “Añadir” el punto pasará a formar parte de la trayectoria cargada y aparecerá en la lista inferior. Si lo que se desea es editar un punto ya cargado de la trayectoria basta con seleccionarlo y la tabla de introducción de datos se actualizará con los valores correspondientes a ese punto. En este momento es posible editar cualquiera de ellos y pulsar sobre el botón “Editar” para que los cambios se guarden y tengan efecto. Bajo la lista de puntos a la derecha se encuentra la opción “Borrar punto”. Este botón permite eliminar cualquier punto seleccionado de la trayectoria cargada.

También bajo la lista de puntos se encuentran las opciones para el manejo de ficheros. Como ya se ha mencionado la interfaz permite ejecutar trayectorias grabadas en ficheros así como guardar trayectorias creadas en ella en el disco duro del ordenador o en cualquier otro medio de almacenamiento extraíble. Los ficheros que la interfaz genera tienen la siguiente forma:

```
-6.420000 5.820000 15.380000 4.660000 4.800000 13.760000
-2.751589 2.494431 6.591813 1.997259 2.057263 5.897487
2333
-1.310000 0.010000 11.970000 3.540000 10.790000 28.570000
-0.721410 0.005507 6.591813 1.949459 5.941994 15.733342
1815
-0.260000 0.450000 9.830000 -2.440000 11.670000 12.440000
-0.174351 0.301762 6.591813 -1.636218 7.825683 8.342031
1491
-1.710000 2.260000 15.570000 -4.580000 2.100000 24.370001
-0.723956 0.956808 6.591813 -1.939018 0.889069 10.317436
2362
```

IMAGEN 29 - EJEMPLO DE FICHERO DE TRAYECTORIA DE MANFRED

Cada tres líneas del archivo definen un punto. En la primera fila se encuentran las posiciones de todas las articulaciones separadas por espacios, en orden descendente desde el hombro hasta el extremo del brazo. La segunda fila contiene, en este mismo orden, la velocidad que tendrá cada articulación en el instante definido en la tercera línea, donde se encuentra el tiempo del punto.

El botón "Cargar archivo" hace aparecer un pequeño apartado con un campo "Examinar…", que permite seleccionar la localización del archivo que se desea enviar entre los discos locales del ordenador, un botón "Enviar" y otro "Cancelar". Cuando se envía un fichero, antes de procesarlo se analiza su forma, y si ésta no coincide con la explicada en el párrafo anterior, el fichero es descartado.

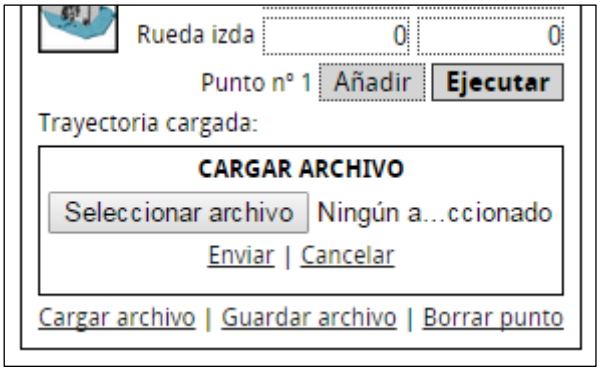


IMAGEN 30 - MANEJO DE FICHEROS EN EL CUADRO DE TRAYECTORIAS

Por último, el botón "Guardar archivo" descargará la trayectoria que se encuentre cargada en ese momento. La localización del archivo dependerá de la configuración del navegador: puede que se consulte al operador o que el fichero se guarde en la ruta por defecto para descargas, el proceso es similar a descargar cualquier archivo de Internet.

4.6.2.5. POSICIÓN

En este apartado se muestran las lecturas de los *encoders* ópticos (ver 3.1.2) acoplados a cada articulación. Para deducir de estas lecturas la posición de cada articulación será necesario realizar la función *homing* (ver 3.1.5), función que puede ser invocada pulsando el botón "HOME" de este cuadro. Cuando el proceso haya terminado los números que se mostrarán corresponden con las desviaciones de esta posición *"home"* o inicial, con lo que pueden considerarse absolutas.

De cada articulación se muestra la posición, la velocidad y el error de posición. Todas estas magnitudes se muestran por defecto en cuentas de *encoder*, aunque pueden ser convertidas a grados utilizando el botón correspondiente bajo la tabla.

Por último, en la parte inferior del control, se muestra la posición del robot en el espacio relativa a la de su arranque. Esta posición está calculada por las funciones de odometría del sistema de procesamiento en ROS, y está formada las coordenadas del punto donde se encuentra Manfred y el ángulo de giro, ambas con respecto al origen definido por la pose en la que se enciende el robot.

4.6.2.6. APRENDIZAJE DE TRAYECTORIAS

El sistema de control de Manfred permite utilizar técnicas de *Programming by Demonstration* (ver 1.2) para el aprendizaje de trayectorias. Este cuadro ofrece los controles necesarios para ejercitar dicha técnica.

Para comenzar hay que hacer clic sobre el botón "Abrir programa". Este botón iniciará el software correspondiente en la tarjeta PMAC que configurará los motores del robot del modo adecuado y esperará nuevas órdenes, mientras monitoriza la posición de las articulaciones. Este botón, a su vez, abrirá un cuadro donde podrá verse un histórico de los comandos enviados a la tarjeta de control, para que el operador sepa constantemente en qué punto de la secuencia de aprendizaje se encuentra:

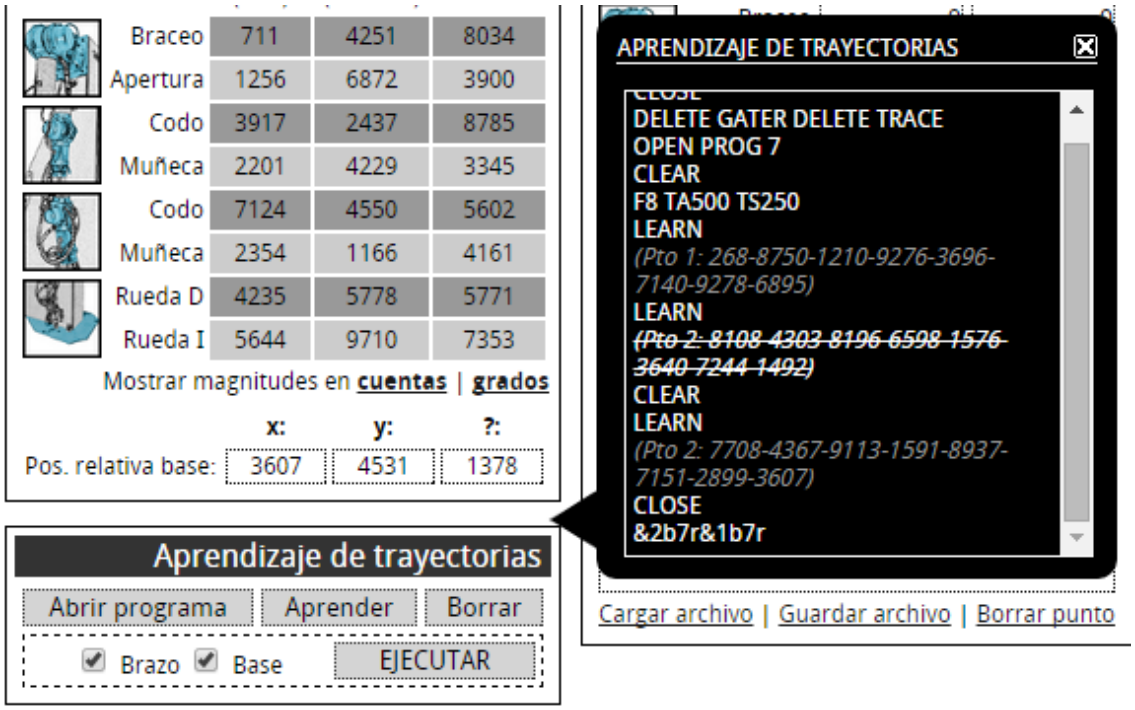


IMAGEN 31 - PROGRAMA DE APRENDIZAJE DE TRAYECTORIAS ABIERTO

Una vez cargado el programa sólo hay que mover el brazo manualmente hasta la posición deseada y pulsar el botón “Aprender” para memorizarlo. El último punto memorizado puede ser eliminado con el botón “Borrar” (el punto será tachado en el histórico para que quede constancia).

Cuando el aprendizaje finalice el botón “EJECUTAR” replicará el movimiento aprendido. Cada posición ejecutada irá cambiando a un tono más oscuro como puede verse en la imagen. En este modo de programación se puede desactivar tanto la base como el brazo, si sólo se desea trabajar con uno de ellos. Dos casillas a la izquierda del botón “EJECUTAR” hacen posible la selección.

Si el globo del programa se cierra en cualquier momento antes de ejecutar el movimiento aprendido el programa de aprendizaje se cierra y los puntos que puedan haber sido aprendidos hasta el momento son descartados. Cerrarlo es, por tanto, un modo de reiniciar el aprendizaje si se desea.

4.7. FUNCIONAMIENTO INTERNO

En el correcto funcionamiento de la interfaz web de Manfred intervienen varios programas o módulos y cada uno desempeña una función esencial. En el siguiente esquema pueden observarse los distintos componentes que participan en la ejecución de órdenes desde la interfaz en el robot y cómo se relacionan entre ellos (nótese el sentido de las flechas, que representan los flujos de datos):

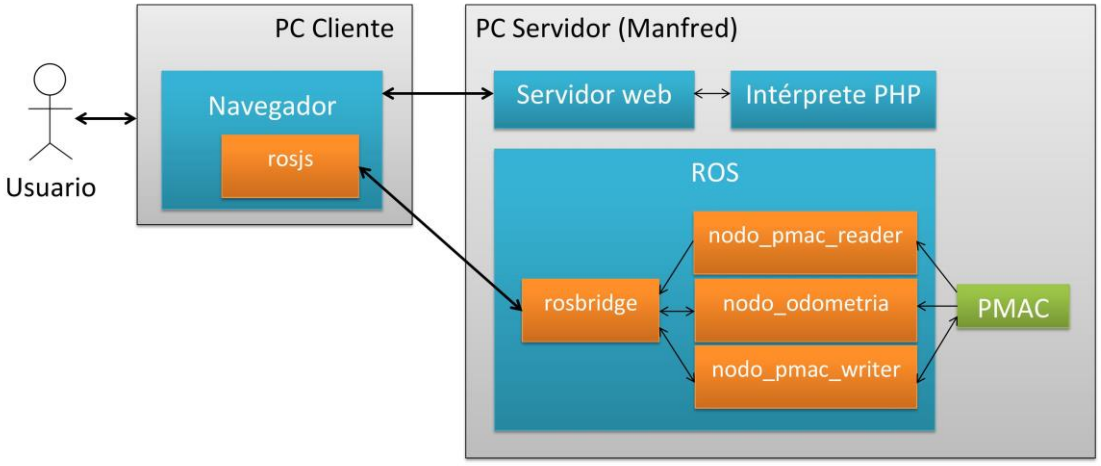


IMAGEN 32 - ESQUEMA DE COMPONENTES DEL CONTROL DE MANFRED

Como puede verse en la imagen, no hay ningún tipo de comunicación entre el servidor Apache y el sistema de control. Esto permite que el servidor pueda alojarse en una máquina diferente a la de Manfred, aunque no se le ha encontrado ninguna aplicación práctica a esta posibilidad. De cualquier forma, para hacerlo sólo hay que escribir la dirección IP del PC con el sistema de control en el fichero de configuración de la interfaz (ver 4.11.1).

Al acceder a la interfaz, el navegador pide y descarga los datos de ésta al servidor web. Entre estos datos se encuentra la librería *rosjs* (ver 4.5), mediante la cual se establece la comunicación entre el sistema de control de ROS (a través del nodo *rosbridge*). Como ya se ha explicado, este nodo permite utilizar los mecanismos de comunicación propios de ROS, haciendo así posible la comunicación de elementos externos a éste, como es el caso de la interfaz.

A través de estos mecanismos, la interfaz se comunica con tres nodos, a saber:

- Las posiciones de los *encoders* son leídas de los registros correspondientes de la tarjeta PMAC por el nodo *nodo_pmac_reader* y publicadas a través de un *topic* llamado */topic_pmac_reader* al que la interfaz se suscribe al conectar. En este *topic* se publica cada 200 ms un array de seis elementos con la posición, la velocidad y error de medida en cuentas de cada articulación. Este nodo hace además la corrección de estos valores después de realizar un *homing*.
- La información de odometría es calculada por el nodo llamado *nodo_odometria*. Ofrece un servicio llamado */srv_odometria* que no tiene ningún argumento de entrada y devuelve una estructura con las

coordenadas de la posición y la orientación relativas del robot desde su arranque. Este *service* será invocado por la aplicación cada 100 milisegundos (aunque este valor es configurable). La respuesta consiste en la "pose" del robot, es decir, su posición y su orientación:

```
geometry_msgs/Pose.msg
   • Point position
        ○ float64 x
        ○ float64 y
        ○ float64 z
   • Quaternion orientation
        ○ float64 x
        ○ float64 y
        ○ float64 z
        ○ float64 w
```

IMAGEN 33 - ESTRUCTURA Y TIPOS DEL MENSAJE "POSE.MSG" DE ROS

La coordenada z de la posición no es relevante, ya que Manfred sólo se mueve en dos dimensiones. La orientación del robot se presenta como un ángulo de giro (más significativo que las cuatro coordenadas de orientación que ofrece el mensaje estándar de ROS) respecto al original. Este ángulo puede deducirse de estos datos utilizando la siguiente ecuación:

$$\theta = \sin^{-1}\Big(-2 \cdot \big((x \cdot z) - (w \cdot y)\big)\Big)$$

donde x, y, z y w pertenecen al cuaterno de la orientación del mensaje.

- Y por último de la recepción y procesamiento de órdenes se encarga el nodo *nodo_pmac_writer*. Este nodo ofrece un *service* llamado */pmac_writer_service* que permite enviar comandos a la tarjeta PMAC (ver 3.1.6). La respuesta de este servicio es la cadena de respuesta devuelta por la tarjeta y un código de error. Esta cadena es la que se imprime bajo la línea de comandos al enviar una orden a través de esta.

Este último servicio, aparte de ser utilizado para escribir en la tarjeta PMAC los comandos introducidos a través de la línea de comandos, también se usa para enviar los comandos que corresponden a las órdenes del resto de controles. Éstos calculan la cadena que corresponda y la envían del mismo modo que si el operador la introdujera a mano. A continuación se muestra un ejemplo del comando para mover un motor un número determinado de cuentas a velocidad uniforme:

```
#[número de articulación]j^[número de cuentas]
```

Para construir órdenes como esta se necesita conocer la numeración que utiliza internamente el sistema de control de Manfred. Por su parte la interfaz, de manera

interna también, numera las articulaciones de 0 a 5 desde el hombro superior hasta la muñeca situada más a la punta. En la siguiente tabla se muestra la correspondencia entre estas dos numeraciones, que puede resultar útil para futuras adaptaciones o modificaciones del control:

Descripción	Numeración interfaz web	Numeración sistema de control
Apertura de hombro	0	4
Braceo de hombro	1	6
Codo intermedio	2	8
Muñeca intermedia	3	5
Codo inferior	4	7
Muñeca inferior	5	3
Rueda derecha	6	1
Rueda izquierda	7	2

Estos datos junto con los parámetros físicos incluidos en los archivos de configuración (ver capítulo 4.11) permiten a la interfaz calcular el comando necesario para, por ejemplo, girar 5 grados la muñeca intermedia:

- El número de articulación correspondiente es el 5.
- Según la configuración (por defecto), un giro de 1 grado del enconder de esta articulación devuelve una lectura de 568.89 cuentas (variable nce5, ver capítulo 4.11.2), por lo tanto para girar 5 grados tendrán que pasar:

$$5° \cdot \frac{568{,}89\ cuentas}{1°} = 2844{,}45\ cuentas$$

Con estos datos ya es posible construir el comando que producirá en el brazo el giro deseado. El número de cuentas ha de ser redondeado al entero más próximo ya que por definición es la mínima medida que el sistema de control puede tomar:

```
#5j^2844
```

La interfaz también genera comandos más complejos, como por ejemplo, los que utiliza para girar la base un número de grados determinado moviendo una rueda hacia delante y la contraria hacia detrás, pero todos se deducen del mismo conjunto de datos como en el ejemplo anterior. En este caso se parte de la fórmula expuesta en el capítulo 3.1.3.

4.8. Modo depuración

La interfaz cuenta con un modo especial de depuración (o *debug*, en inglés) que muestra ciertos valores internos que afectan al comportamiento de la interfaz sin ser

mostrados explícitamente. En este modo también pueden analizarse los comandos enviados al sistema de control y las respuestas que se obtienen de éstos, de manera que puede apreciarse el funcionamiento "trasero" del control.

Para activar este modo hay que definir mediante el método GET del protocolo HTTP una variable llamada "debug". Esto se hace añadiendo la cadena "?debug" a la URL de la interfaz, como en el siguiente ejemplo en el que el robot tiene la dirección IP 192.168.56.101:

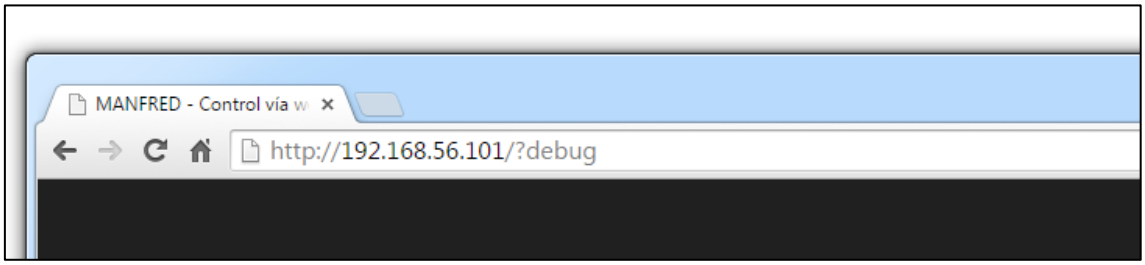


IMAGEN 34 - ENTRADA EN MODO DEPURACIÓN O *DEBUG*

Cuando este modo está activo, en la parte inferior izquierda de la pantalla de control aparece un pequeño menú con tres opciones. La primera de ellas despliega al pulsarla un cuadro dónde es posible analizar el array $_POST de PHP, que contiene todas las variables que han sido enviadas a través del método POST de HTTP y sus valores. Este método es el utilizado por el control de acceso, por ejemplo:

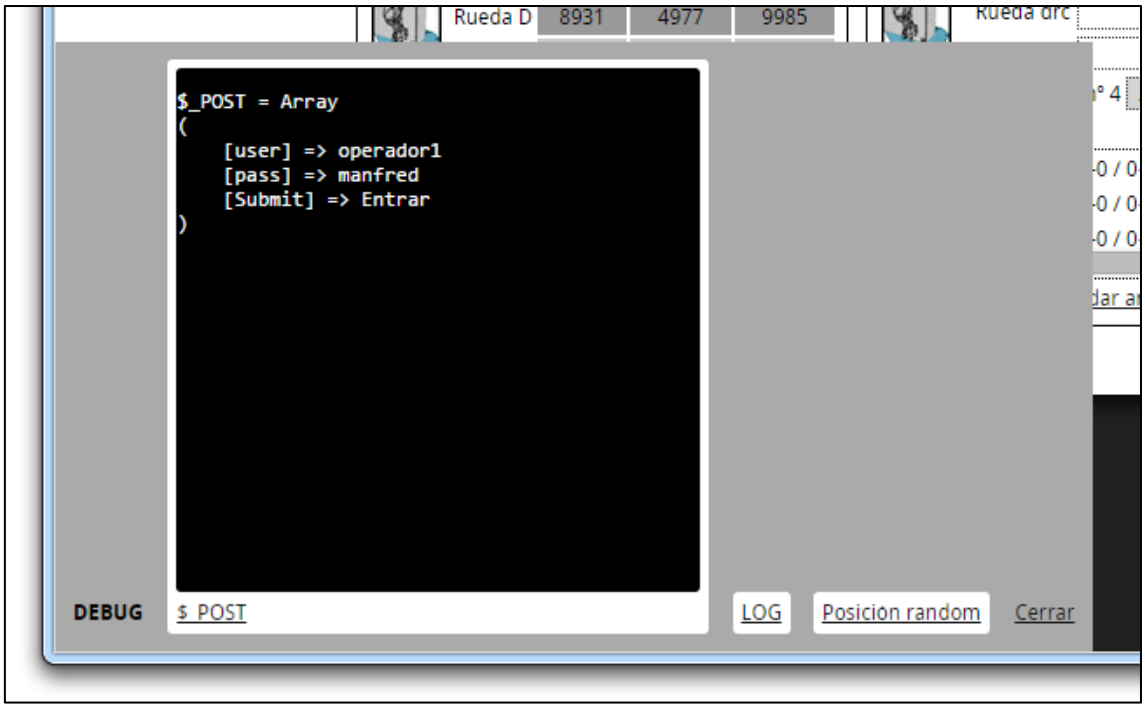


IMAGEN 35 - CUADRO $_POST DEL MODO DE DEPURACIÓN

La segunda opción permite mostrar un log donde aparecen distintas acciones internas, como los intentos de conexión con el sistema de control y su resultado; y las llamadas a servicios o los datos recibidos a través de los *topics* a los que la interfaz está suscrita:

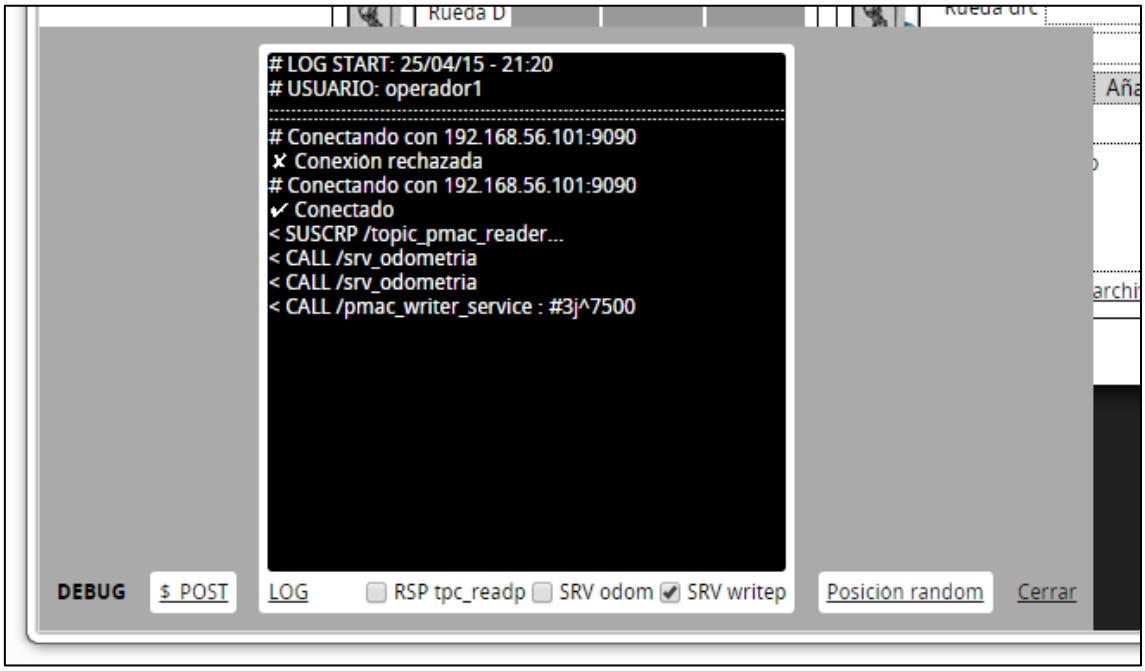


IMAGEN 36 - LOG DEL MODO DEBUG

Los tres cuadros de selección que aparecen bajo las líneas del log permiten filtrar la información que aparece en éste, de manera que es posible mostrar sólo la información que se quiera analizar:

- RSP tpc_readp: si esta opción está activa, se muestran las respuestas al *topic* que devuelve los valores de posición de los encoders del brazo.
- SRV odom: mostrar o no las llamadas al servicio de odometría y las respuestas que este proporciona.
- SRV writep: igual que la opción anterior, pero para el servicio */pmac_writer_service*.

Por último, la tercera opción activa una función JavaScript que actualiza con números aleatorios todas las posiciones del cuadro de control. Esto facilita la depuración (gráfica sobretodo) de este cuadro cuando no es posible una conexión con el sistema de control de Manfred.

4.9. REQUISITOS – COMPATIBILIDAD

Para ejecutar la interfaz en un ordenador sólo es necesario un navegador compatible con Javascript pero, como se explicó en el capítulo 2.3, no todos los navegadores representan de la misma forma las páginas web. La interfaz se ha desarrollado manteniendo una compatibilidad total con los navegadores Google Chrome y Mozilla Firefox, ambos ampliamente utilizados y comprometidos y fieles con los estándares más extendidos. Estos navegadores están además disponibles para varios sistemas operativos (Windows, iOS y Linux). Existen fallos de compatibilidad conocidos con Internet Explorer provocados porque éste no se ciñe a los estándares web. De cualquier forma, la interfaz también es utilizable en este navegador.

Los requisitos mínimos de hardware del ordenador dependerán de los del navegador que se vaya a utilizar, a continuación se muestran los de los dos recomendados:

Google Chrome	Mozilla Firefox
• Procesador: Pentium 4 • RAM: 512 MB • Espacio en disco: 350 MB	• Procesador: Pentium 4 • RAM: 512 MB • Espacio en disco: 200 MB

4.10. ESTRUCTURA DEL PROGRAMA (CÓDIGO)

El código de la interfaz web de Manfred se estructura de la siguiente forma:

- Directorio raíz: Se trata de la carpeta que el servidor web utiliza como base de los archivos que va a servir. En esta carpeta se encuentran:
  - El fichero index.php, que es el punto de acceso a la interfaz. Es el primer fichero que leerá y enviará el servidor Apache al navegador al acceder a la interfaz.
  - Los ficheros de configuración config.js y config.php. Se encuentran en la raíz para que sea más sencillo editarlos. Su contenido se encuentra analizado en el siguiente apartado.
  - El fichero trayectoria.txt, en el que se almacena la trayectoria en edición por el cuadro Trayectorias de la interfaz.
  - El resto de carpetas en las que se organizan los ficheros que componen la interfaz.
- Directorio "cont": esta carpeta contiene los ficheros .php que describen cada parte de la interfaz. Dentro se pueden encontrar:
  - El fichero control.php que codifica la pantalla principal de la interfaz, donde se encuentran el resto de controles.
  - El fichero login.php, que describe la pantalla de login.

    - El fichero loginM.php, similar al login.php pero este contiene una versión más reducida optimizada para pantallas pequeñas.
    - La carpeta “bloques”. Esta carpeta contiene un fichero .php por cada cuadro de control que se puede encontrar en la pantalla principal de la interfaz. Estos ficheros son incluidos en la posición correspondiente por el archivo control.php.
- Directorio “fonts”: dentro se encuentran los tipos de letra que se utilizan en la interfaz. Incluyendo estos ficheros se consigue que la interfaz se muestre correctamente aunque el ordenador desde el que se acceda no tenga esas fuentes instaladas.
- Directorio “img”: contiene los archivos de imagen de la interfaz, por ejemplo, el esquema del robot que aparece en el control simple, las miniaturas de cada articulación que hay en el resto de controles o el fondo de la pantalla de login.
- Directorio “includes”: en esta carpeta están guardadas todas las librerías que utiliza la interfaz, ya sean de ejecución en servidor o en cliente. Contiene librerías creadas específicamente para la interfaz así como librerías y bibliotecas externas, como por ejemplo *rosjs* o jQuery. Los archivos JavaScript están agrupados en una subcarpeta llamada “js” porque la interfaz cuenta con una función que agrupa todos los ficheros que hay en esta carpeta y los envía al navegador en una sola petición, con el consiguiente ahorro de tiempo de carga.

4.11. CONFIGURACIÓN

Hay varios valores de la interfaz que se han hecho configurables para adaptarla mejor a los distintos procedimientos que vayan a realizarse con ella, o para hacer fácil un cambio en caso de modificar algún componente de Manfred. Los datos configurables están divididos en dos ficheros, uno con parámetros configurables de la interfaz y otro con valores que dependen de los elementos del robot, tales como dimensiones o características de sensores, por ejemplo.

A continuación se describen la estructura y el contenido de estos dos ficheros, para hacer posible su posterior modificación en caso de necesidad.

4.11.1. ARCHIVO DE CONFIGURACIÓN DE LA INTERFAZ

Como se acaba de mencionar, este fichero contiene variables que permiten modificar ciertos aspectos de la interfaz, tales como el tiempo máximo de sesión (mencionado en el capítulo 4.6.1) o la lista de operadores y sus correspondientes contraseñas de acceso.

Se trata de un fichero PHP, con lo que debe atenerse a sus normas sintácticas: las variables comienzan con el símbolo del dólar ($), las cadenas de caracteres se escriben entre comillas simples (‘) o dobles (“”) y todas las líneas que comiencen por dos barras inclinadas (//) o una almohadilla (#) son consideradas comentarios por el intérprete y no se tienen en cuenta. Cada orden debe terminar por un símbolo de punto y coma (;).

A continuación puede verse un ejemplo de este fichero, con los valores por defecto y comentado:

```
<?php
//////////////////////////////////////////////////////////////////////
//
// CONTROL VIA WEB ROBOT MANFRED
// José Luis Ortiz Navarro
// UC3M
//
// SCRIPT HOST - PHP
//
// config.php
// Parámetros de configuración
//
//////////////////////////////////////////////////////////////////////


#########################################################################
# Tiempo en minutos para timeout de login
#########################################################################
$timeout = 30;

#########################################################################
# Array de usuarios/passwords
# $users['USUARIO'] = 'PASSWORD ENCRIPTADO EN MD5';
# ó bien
# $users['USUARIO'] = md5('PASSWORD');
#########################################################################
$users['pepe'] = 'a52bde8a9926723d693745340c8f4731'; //manfred1
$users['juan'] = 'c893bad68927b457dbed39460e6afd62'; //prueba

#########################################################################
# Dirección IP del robot desde el cliente
# $_SERVER[SERVER_ADDR] si el servidor está en el robot
#########################################################################
#$IP_robot = 'localhost'; // Pruebas
$IP_robot = $_SERVER[SERVER_ADDR]; // Uso

#########################################################################
# Puerto de comunicación con ROSBRIDGE
# Por defecto: 9090
#########################################################################
$RB_port = '9090';

#########################################################################
# Archivo donde se almacena la trayectoria
# La ruta es relativa a la raiz del servidor
#########################################################################
```

```
$archTray = 'trayectoria.txt';

?>
```

IMAGEN 37 - EJEMPLO DE FICHERO DE CONFIGURACIÓN DE LA INTERFAZ

Esta es la lista de variables que este fichero debe contener obligatoriamente para el correcto funcionamiento de la interfaz, junto con el tipo y la descripción de cada uno:

Nombre	Tipo	Descripción
timeout	Número	Tiempo máximo de la duración de una sesión iniciada por un operador en la interfaz, en minutos. Cuando este tiempo pasa la interfaz sigue funcionando con normalidad, solo que será necesario volver a registrarse en caso de cerrar el navegador o actualizar la página.
users	Diccionario de cadenas	Lista de usuarios y sus respectivas contraseñas de acceso. Cada línea corresponde a un nuevo operador. El índice del diccionario (la cadena que se encuentra entre corchetes) corresponde al nombre que el operador deberá introducir. El valor que se le asigna a este elemento es la contraseña codificada en MD5 (esto permite que las contraseñas no estén escritas directamente en el fichero de configuración y sea muy difícil averiguarlas en caso de acceder a él). En el fichero de ejemplo se encuentran comentadas las contraseñas antes de ser codificadas, pero esta práctica no es recomendable.
IP_robot	Cadena	Dirección IP del robot Manfred. Si el servidor Apache (y el intérprete PHP) se encuentran en el mismo PC en el que corre ROS, esta variable puede ser definida automáticamente asignándole el valor de la variable global *$_SERVER[SERVER_ADDR]* (sin comillas), como en el ejemplo. Esta variable contiene la IP del servidor donde se está ejecutando el intérprete, que en este caso será la misma que la del sistema. Si en cambio el servidor web se ejecuta en un equipo distinto al del sistema de control esta variable tendrá que contener la dirección IP de éste.
RB_port	Cadena	Puerto TCP al que está conectado el nodo *rosbridge*. Se trata de un valor configurable el dicho nodo, simplemente deberán coincidir. Aunque se trate de un valor numérico debe ser introducido como una cadena, es decir, entre comillas.
archTray	Cadena	Fichero donde se almacenan las trayectorias en edición por la interfaz. Modificar esta variable no supone ningún cambio funcional, el único requisito que debe cumplir es apuntar a un fichero que sea escribible por el usuario con el que se ejecuta el intérprete PHP.

4.11.2. ARCHIVO DE CONFIGURACIÓN DE PARÁMETROS DEL ROBOT

En este archivo contiene ciertos parámetros que miden características físicas del robot susceptibles de cambiar en el futuro, de forma que sea sencillo reconfigurar

la interfaz en ese caso. Un claro ejemplo de esto sería sustituir un *encoder* por otro con una resolución mayor para conseguir movimientos más precisos.

En este caso el fichero tiene formato JavaScript, y como el anterior, éste debe ajustarse a su sintaxis, al menos a las normas que afectan a este fichero: las líneas que comienzan con dos barras inclinadas (//) se consideran comentarios y el intérprete las obvia; las variables se declaran con la etiqueta "var" seguida del nombre de la variable y cada orden se termina con un símbolo de punto y coma (;), aunque entre declaraciones de variables no es necesario y en el siguiente ejemplo se han eliminado por simplificarlo:

```
////////////////////////////////////////////////////////////////////////
//
// CONTROL VIA WEB ROBOT MANFRED
// José Luis Ortiz Navarro
// UC3M
//
// SCRIPT CLIENTE - JAVASCRIPT
//
// config.js
// Valores configurables para el manejo del robot. Constantes en tiempo
// de ejecución
//
////////////////////////////////////////////////////////////////////////

// Periodo (en milisegundos) entre llamadas al servicio de odometría
// (consulta de la posición de la base
var intervalPosBase = 100

// CONFIGURACIÓN DE LA BASE
// Numero de cuentas total de los encoder de la base
var nceb = 500
// Reduccion motores de la base
var rmb = 20
// Diametro de la base (en centímetros)
var db = 48.5
// Diametro de las ruedas de la base (en centímetros)
var dr = 15.67145673

// CONFIGURACIÓN DEL BRAZO
// Número de cuentas por grado para el encoder de cada motor
var nce3 = 750
var nce4 = 1820.44
var nce5 = 568.89
var nce6 = 1820.44
var nce7 = 682.67
var nce8 = 1820.44
```

IMAGEN 38 - EJEMPLO DE FICHERO DE PARÁMETROS DEL ROBOT

Este fichero debe contener al menos todas las variables incluidas en la siguiente tabla, donde también se indica su tipo y una pequeña descripción del efecto de cada una:

Nombre	Tipo	Descripción
intervalPosBase	Número	Periodo de actualización de los datos de posición odométrica de Manfred. Determina la frecuencia con la que el servicio */srv_odometria* es llamado (ver 4.7). Expresado en milisegundos.
nceb	Número	Número de cuentas por revolución completa de los *encoders* conectados al eje de cada rueda de la base del robot.
rmb	Número	Relación de giro entre los motores y las ruedas del robot.
db	Número	Distancia entre las ruedas motrices del robot, expresada en centímetros.
dr	Número	Diámetro de las ruedas motrices del robot, en centímetros.
nce[3…8]	Número	Número de cuentas o pulsos que devuelve cada *encoder* por cada grado sexagesimal que gira. El número corresponde a la articulación con la que el *encoder* gira solidario, según la numeración del robot (ver 4.7).

Capítulo 5. Resultado

Este proyecto ha dado como resultado una interfaz intuitiva y fácil de usar; fácilmente accesible (desde varios tipos de dispositivo), interactiva, inalámbrica e integrada con el software previamente existente en el robot. Ha resultado en una forma sencilla y rápida de manejar el manipulador, sin la necesidad, como ocurría antes, de abrir un escritorio remoto o conectar algún dispositivo de entrada (teclado y/o ratón) al cuerpo del robot. Además, puede utilizarse desde distintos navegadores de escritorio, pero también desde distintos dispositivos, como una tablet o un teléfono inteligente.

Al ser una aplicación el producto de la elaboración de este proyecto no pueden incluirse más resultados que las capturas de pantalla que ilustran los capítulos anteriores.

5.1. Ejemplos de uso

El empleo de esta interfaz puede ser conveniente durante el desarrollo diferentes tareas de investigación o mantenimiento del robot. Sin necesidad de establecer conexiones específicas, incluso sin utilizar un ordenador, es posible ahora enviar comandos rápidos al robot o consultar las lecturas de sus sensores de un simple vistazo. A continuación se desarrollan algunos ejemplos en los que el uso de la interfaz acelera y simplifica algunas actividades que se realizan de manera frecuente con Manfred.

5.1.1. Cambio o recalibración de sensores

Si cualquiera de los sensores del brazo manipulador es sustituido, por otro de mayor precisión por ejemplo, varios cambios en el software del robot serán necesarios: instalación de drivers, modificación de factores de corrección, etcétera. El cuadro de posiciones con el que cuenta la interfaz permite, en primer lugar, comprobar si las lecturas directas (las cuentas) que se están recibiendo del sensor son

correctas o hay algún problema. En segundo lugar también es posible comprobar si las transformaciones que se hacen sobre este dato se han calculado de forma correcta, gracias a que los resultados son mostrados de forma clara y en tiempo real.

El botón de emergencia KILL ALL hace también posible probar de una forma sencilla la sustitución de alguno de los sensores capacitivos utilizados para la función HOME. Si el nuevo sensor falla durante las pruebas en la detección de la referencia que detiene de forma segura el movimiento de calibración y el motor correspondiente continúa girando es muy probable que se originen daños graves en el manipulador o en cualquier objeto en el área de alcance de éste. La presencia de este botón permite una detención de emergencia de todos los motores minimizando el riesgo (ver 4.6.2.3).

5.1.2. Pruebas de nueva funcionalidad

La línea de comandos presente en la interfaz desarrollada resulta una forma muy rápida de ejecutar comandos que no están implementados en ningún control de la aplicación: es posible enviar órdenes al sistema de control del robot (la parte de software que funciona sobre ROS) que no han sido contempladas hasta el momento. Esto resultará muy útil durante la ejecución de los futuros desarrollos de Manfred ya que permitirá rápidamente probar comandos recién incluidos en el código de la tarjeta PMAC mientras se observa, por ejemplo, la posición y velocidad instantáneas de todas las articulaciones.

De la misma forma que en el caso de la sustitución de un sensor capacitivo, cualquier nueva función que se pruebe en Manfred puede generar situaciones de riesgo propio o externo, ya que se trata de movimientos de motores con cierta potencia acoplados a reducciones importantes. Este riesgo será más elevado cuanto más tempranas sean estas pruebas o ejecuciones. El botón KILL ALL permite también en este caso detener inmediatamente todos los motores evitando, o al menos minimizando los posibles daños.

Además la versatilidad de la interfaz permite, por ejemplo, tenerla abierta en un teléfono móvil conectado a la red Wi-Fi mientras se trabaja en el desarrollo y la depuración de la nueva funcionalidad en cuestión en un ordenador de escritorio o en un portátil, de forma que el botón esté siempre accesible.

5.1.3. Traslados

Gracias a la interfaz es posible trasladar a Manfred de una forma muy sencilla de habitación o incluso de edificio. El uso de la tecnología Wi-Fi permite hacerlo incluso por zonas dónde no exista cobertura de este tipo, ya que estos dispositivos pueden crear por si solos redes a las que es posible conectarse desde un teléfono o

cualquier otro aparato compatible, simplemente configurando convenientemente el adaptador en el sistema operativo. Estas redes son conocidas como redes *ad-hoc*.

Ya sea mediante este método o utilizando infraestructura existente (una red corporativa de amplio alcance como la de la Universidad, por ejemplo), mover sin cables al robot es tan fácil como colocar su selector de alimentación en modo baterías, abrir la interfaz, seleccionar la base en el esquema del control simple y manejarlo con las flechas.

5.2. FUTURAS IMPLEMENTACIONES

Ciertas partes del código han sido preparadas para el desarrollo de nuevas funcionalidades o mejoras de las ya existentes. Este pre acondicionamiento ha surgido en ocasiones fruto del propio desarrollo del código, al por ejemplo aparecer casos a considerar que actualmente no tienen sentido; otras veces ha sido escrito a propósito para facilitar la incorporación de futuros fragmentos de código encargados de permitir la teleoperación de los nuevos desarrollos de Manfred.

5.2.1. TRAYECTORIAS CON MOVIMIENTOS DE LA BASE

Actualmente el sistema de control de Manfred no admite trayectorias que involucren a la base, sólo a las articulaciones del brazo manipulador. Al desarrollar la función Javascript que envía de forma asíncrona las órdenes correspondientes para ejecutar la trayectoria cargada hubo que crear una estructura que asignara a cada articulación del robot un carácter que utiliza el sistema de control para identificarlas. Fue en este momento cuando surgió la posibilidad de añadir dos posiciones más a este diccionario y preparar el resto de la función para esta mejora:

```
[…]
    case "ejecutar":
      // La ejecución es por JS, no hay que enviar el formulario:
      var envio;
      flagError = 1

      comando_srvCall('CLOSE');
      comando_srvCall('DELETE GATER DELETE TRACE');
      comando_srvCall('OPEN PROG 5');
      comando_srvCall('CLEAR');
      comando_srvCall('INC');

      // Traduce nros. de artic web a letras de trayectoria
      // Las dos últimas no se utilizan, futura implementación
      var dicWRlit = ['Z','A','B','C','U','W','X','Y'];

      // Bucle for para recorrer cada punto
      for (i = 0; i < trayectoria.length; i++) {
        comando_srvCall('PVT('+trayectoria[i][2]+')');
        envio = '';

        // Bucle for para recorrer cada componente del punto
        // Para añadir ruedas cambiar el 5 por 7, lo demás está todo hecho
        for (j=0; j <= 5; j++) {
          envio += dicWRlit[j]+trayectoria[i][0][j]+':'+trayectoria[i][1][j]+' ';
        }
```

```
        envio = envio.substring(0,envio.length-1)
        comando_srvCall(envio);
      }

      comando_srvCall('CLOSE');
      comando_srvCall('&1b5r&2b5r');
      break;
  }
```

IMAGEN 39 – FRAGMENTO DEL CÓDIGO DEL ENVÍO DE TRAYECTORIAS

Este fragmento de código se encuentra en el fichero '7.trayec.js'. Como pone en el último comentario que puede verse en el fragmento de código, cuando el sistema de control esté preparado para ejecutar trayectorias con movimientos de la base simplemente habrá que cambiar el 5 del bucle que tiene debajo por un 7.

5.2.2. ACCIONES PARA LA PINZA

El control simple permite seleccionar una pinza situada al extremo del brazo antropomórfico de Manfred. Al seleccionarla se muestran dos botones con los rótulos "Abrir" y "Cerrar", pero ninguno de los dos tiene asociada funcionalidad alguna, ya que actualmente esa parte del sistema de control del robot no se encuentra finalizada. Estos botones sirven como ejemplo para el momento en el que sea necesario añadir acciones a esta pinza o a otros elementos que puedan ser acoplados a la punta. A continuación se muestra el fragmento de código del fichero '4.simple.js' donde habría que introducir las funciones asociadas a estos botones:

```
[…]
function movSimple(direccion) {
  if (typeof($('.sel_artic:checked').val()) == 'undefined') {
    window.alert("Primero hay que seleccionar una articulaci\xF3n");
  }
  else if ($('#paso').val() == ""  || (isNaN($('#paso').val()))) {
    window.alert("No hay valor de paso o no es num\xE9rico");
  }
  else {
    ejecMovIncremental(direccAmov($('.sel_artic:checked').val()+direccion))
  }
}
```

IMAGEN 40 - CÓDIGO ASOCIADO A LAS FUNCIONES DE LA PINZA

Una nueva rama en el *if* condicionada a que el valor de la variable 'direccion' sea igual a las cadenas "abr" o "crr" permitiría codificar estas acciones.

5.2.3. FUNCIONES ASOCIADAS A LA FUNCIÓN *HOMING*

En el fichero '8.posic.js', dónde se encuentran las funciones asociadas al cuadro de posiciones, puede encontrarse el siguiente fragmento de código, que pertenece a la función ejecutada al pulsar el botón "HOME":

```
[…]
function home() {
  //DESACTIVADO
  /*
```

```
  var i;
  for (i=0; i<=5; i++) {
    $("#mov"+i+"_abs").removeAttr('disabled');
  } */
  comando_srvCall('&2b1r');

}
```

IMAGEN 41 - FRAGMENTO DEL CÓDIGO DE LA FUNCIÓN *HOMING*

Bajo el comentario "DESACTIVADO" se encuentra (comentado también) un ejemplo de acción que podría asociarse a esta función: habilitar un nuevo selector con el que contaría cada articulación para moverla hasta una posición absoluta. Pero como ya se ha dicho esto es sólo un ejemplo, puede añadirse cualquier otra función antes o después de la llamada a la tarjeta PMAC que desencadena la calibración (la última orden en el fragmento superior).

5.2.4. NUEVA VERSIÓN DE ROSBRIDGE (2.0)

En el momento de la finalización de este proyecto se encuentra disponible una nueva versión del paquete *rosbridge* acompañada de una nueva biblioteca Javascript para comunicarse con él. Su lanzamiento ocurrió pasado el cierre del código de la interfaz, y sería necesario actualizar ROS por completo ya que sólo es compatible con versiones superiores a *Fuerte Turtle*, pero no debería ser demasiado complicado actualizarla para utilizar los nuevos recursos.

Esta afirmación se apoya en el hecho de que todas las llamadas a la biblioteca *rosjs*, que al final son las que habría que sustituir o modificar, se encuentran reunidas en el mismo fichero: '2.conex.js'. Al comienzo de este fichero pueden encontrarse las funciones encargadas de establecer la conexión con *rosbridge*, así como de vigilar e informar de cualquier error que ocurra en el transcurso de esta.

Pero la ventaja del código ante esta posible actualización viene a continuación: se definen tres funciones para interactuar con los tres nodos de ROS mencionados en el capítulo 4.7:

```
[…]
// Manejador para las respuestas del topic /topic_pmac_reader que
// informa de la posición de cada articulación del brazo

function posBrazo_topicHandler() {
  var nombreTopic = '/topic_pmac_reader';
  if (debug == true) {
    log('< SUSCRP '+nombreTopic+'...');
  }
  connection.addHandler(nombreTopic,function(msg) {
    var i;
    for (i=0; i<=7; i++) {
      actualizPosBrazo(i,0,msg.position[dicWR[i]-1]);
      actualizPosBrazo(i,1,msg.velocity[dicWR[i]-1]);
      actualizPosBrazo(i,2,msg.effort[dicWR[i]-1]);
    }
    if (debug == true && $('#dbg_readp').attr('checked') == 'checked') {
      log('> TPC '+nombreTopic+' RCV: '+msg);
```

```
    }
  });
  connection.callService('/rosjs/subscribe','["'+nombreTopic+'",0]',function(e) {
    if (debug == true) {
      log('> SUSCRIP OK '+nombreTopic+'...');
    }
  });
}

// Esta función llama al servicio /srv_odometria y maneja la respuesta
// que devuelve éste: la posición relativa de la base

function posRelatBase_srvCall() {
  var nombreSrv = '/srv_odometria';
  if (debug == true && $('#dbg_odom').attr('checked') == 'checked') {
    log('< CALL '+nombreSrv);
  }

  connection.callService(nombreSrv,'{
      "header": {
         "seq": 0,
         "stamp": {
           "secs": 0, "nsecs": 0
         },
         "frame_id": "s"
      }}',function(rsp) {
    //CALLBACK
    var pose = rsp.odometria.pose.pose;
    actualizPosBase(0,pose.position.x);
    actualizPosBase(1,pose.position.y);
    var orient = pose.orientation;
    var tetha = Math.asin(-2*((orient.x*orient.z)-(orient.w*orient.y)));
    actualizPosBase(2,tetha);
    if (debug == true && $('#dbg_odom').attr('checked') == 'checked') {
      log('> SRV '+nombreSrv+' RSP: '+rsp);
    }
  });
}

// Esta función llama al servicio /pmac_writer_service y le envía
// un comando. Pasa la respuesta a la función log()
// orden: String con el comando a enviar

function comando_srvCall(orden) {
  var nombreSrv = '/pmac_writer_service';

  if (debug == true && $('#dbg_writep').attr('checked') == 'checked') {
    log('< CALL '+nombreSrv+' : '+orden);
  }
  connection.callService(nombreSrv,'{"send": "'+orden+'"}',function(rsp) {
    //CALLBACK
    if (rspComando == true) {
      devolverRSP(rsp.answer);
      rspComando = false;
    }
    if (debug == true && $('#dbg_writep').attr('checked') == 'checked') {
      log('> SRV '+nombreSrv+' ANSW: '+rsp.answer);
      log('> SRV '+nombreSrv+' ERR: '+rsp.error_code);
    }
  });
}
```

IMAGEN 42 - CÓDIGO DE COMUNICACIÓN CON ROS

Estas funciones podrían considerarse alias de las propias de la biblioteca *rosjs*, aunque esto no es exactamente así porque poseen algo de funcionalidad añadida. La ventaja reside en que en el resto del código son estas tres funciones las que son llamadas para comunicarse con el sistema de control, con lo que para realizar el cambio a la nueva versión de la librería no sería necesario tocar ningún otro fichero.

Capítulo 6. CONCLUSIONES

Tras el desarrollo y documentación de la interfaz puede concluirse que se han conseguido cumplir de forma satisfactoria todos los objetivos que se plantearon al comienzo de este proyecto, y que están recogidos en el capítulo 1.4. Además, al haberse diseñado en estrecha colaboración con los usuarios habituales del robot, han podido introducirse ciertas mejoras que sólo pueden ser producto de un uso continuado y prolongado de él.

La teleoperación de Manfred a través de la interfaz es totalmente inalámbrica siempre, claro, que el dispositivo desde el que se accede cuente con tecnología Wi-Fi (lo cual como se ha visto en el capítulo 2.1.1 es más que probable), con lo que el primer objetivo está cumplido.

El segundo objetivo requisita que la interfaz sea accesible desde distintos sistemas operativos y tipos de dispositivo, cuantos más mejor. El hecho de haber sido diseñada en forma de aplicación web maximiza este hecho, ya que todos los sistemas operativos más extendidos, tanto para ordenador como para *smartphone* o *tablet*, cuentan con navegadores web en los que se puede utilizar. Incluso terminales móviles anteriores a la "explosión" de los *smartphones* cuentan con estos navegadores.

Por último, el robot puede ser controlado a través de la interfaz gracias a que ésta funciona junto al sistema de procesamiento diseñado previamente en el sistema operativo para robótica ROS, que es el que se encarga realmente de realizar las tareas de control sobre los movimientos que se efectúan. Con lo cual el tercer objetivo también puede darse por conseguido.

Pero además de cumplir estos tres objetivos, el diseño de esta interfaz ha intentado dar un paso más incluyendo funciones y atajos que la hacen más manejable y versátil. Mejoras como la carga de ficheros de trayectorias o la posibilidad de controlar a Manfred de forma sencilla con las flechas del teclado dan un valor añadido

a la aplicación, así como un elaborado diseño que consigue que su uso sea intuitivo sin perder de vista la estética.

Con todo esto, este proyecto puede darse por concluido satisfactoriamente.

6.1. FUTURAS LÍNEAS DE TRABAJO

A pesar que ser ya totalmente funcional, la interfaz desarrollada es muy susceptible de ser ampliada con nuevas funciones. No hay que olvidar que Manfred es un prototipo de investigación que continúa siendo desarrollado y mejorado en el presente.

Nuevas capacidades como un sistema de visión, o mejoras como un nuevo acoplamiento en la punta del brazo que permita conectar distintos elementos a él están siendo desarrolladas actualmente y pueden ser fácilmente incluidas dentro de la interfaz. El puente de comunicaciones creado por el dúo *rosbridge*-*rosjs* permite que el intercambio de datos entre la interfaz y el sistema de control sea transparente al desarrollador, con lo que para incorporar cualquier ampliación de éste sólo será necesario añadir un nuevo control a la interfaz (tarea que se facilita basándose en el código de los controles ya existentes) donde exponer o recoger la información que tenga que tratarse y codificar las funciones necesarias para transformar o procesar estos datos. Del otro lado, habrá que crear nuevos *topics* o *services* que permitan el envío de órdenes o la recepción de información, posiblemente ofrecidos por los nuevos nodos que se encargarían de la funcionalidad añadida al sistema de control.

Además, considerando especialmente el caso del sistema de visión, pero igualmente válido para cualquier otra funcionalidad en la que se transmitan imágenes, el formato JSON en el que se preparan los datos en la interfaz (ver capítulo 4.5) admite varios tipos de datos, entre los que se encuentran formatos de vídeo e imagen.

ANEXO A. INSTALACIÓN PASO A PASO

En el momento de la realización de este proyecto, la interfaz web de Manfred se encuentra instalada y funcionando en el sistema operativo del robot, pero puede que sea necesario volverla a configurar en caso de actualizarlo o cambiarlo. En este anexo se encuentra una guía que pretende facilitar este proceso mediante unas instrucciones sencillas paso a paso:

1. Instalar el servidor Apache junto con el intérprete de PHP en el sistema. Una forma sencilla (mencionada en el capítulo 4.3) es instalar el módulo de PHP para Apache "libapache2-mod-php5". Este paquete depende de los dos anteriores, con lo cual aceptando los cambios propuestos por el gestor de software se instalará automáticamente todo lo necesario. En Ubuntu, u otro sistema operativo basado en Debian, puede hacerse a través del terminal con el siguiente comando:

```
manfred@manfred:~$ sudo apt-get install libapache2-mod-php5
```

2. Tras la instalación, acceder a http://localhost y comprobar si se muestra la página Editar convenientemente los archivos de configuración. El capítulo 4.11 detalla estos ficheros.
3. Borrar el contenido de la carpeta /var/www y copiar en ella todos los ficheros de la interfaz. Estas operaciones hay que realizarlas con permisos de administrador ya que éste es el dueño de la carpeta. A continuación, proporcionar permisos de acceso público a todos los archivos copiados. Aquí se muestra un ejemplo de cómo asignar estos permisos desde un terminal:

```
manfred@manfred:~$ cd /var/www
manfred@manfred:/var/www$ sudo chmod -R 755 *
manfred@manfred:/var/www$ sudo chmod -R 777 trayectoria.txt
```

4. Presuponiendo que ya se encuentran instalados correctamente ROS y el sistema de control, instalar el paquete *rosbridge*. Este paquete se encuentra en el *stack* de ROS *brown-remotelab* que en Ubuntu se instala mediante el siguiente comando:

```
manfred@manfred:~$ sudo apt-get install ros-electric-brown-remotelab
```

BIBLIOGRAFÍA

- Álvarez, D. (2011). Controlador cartesiano para el brazo LWR-UC3M-1 del robot manipulador MANFRED con detección de contacto.
- *Aplicación web*. (19 de Oct de 2014). Recuperado el 2 de Ene de 2015, de Wikipedia: http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web
- *Automated guided vehicle*. (25 de Nov de 2014). Recuperado el 27 de Nov de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Automated_guided_vehicle
- Blanco, D., Ansari, S. A., Castejón, C., López Boada, B., & Moreno, L. E. (2005). Manfred: Robot antropomórfico de servicios fiable y seguro para operar en entornos humanos. *Revista Iberoamericana de Ingeniería Mecánica, 9*(3), 33-48.
- *Bluetooth*. (14 de Dec de 2014). Recuperado el 2 de Ene de 2015, de Wikipedia: https://en.wikipedia.org/wiki/Bluetooth
- Bluetooth Special Interest Group. (s.f.). *Bluetooth Fast Facts*. Recuperado el 28 de Nov de 2014, de http://www.bluetooth.com/Pages/Fast-Facts.aspx
- Carmona, J. A. (2013). *Androides del pasado, recordando lo que una vez triunfó: HTC Magic*. Recuperado el 08 de Dic de 2014, de andro4all: http://andro4all.com/2013/05/androides-del-pasado-htc-magic
- DELTA TAU Data Systems, I. (2004). *PMAC/PMAC2 Software Reference Manual.* DELTA TAU Data Systems, Inc.
- Fundación Telefónica. (2014). *La Sociedad de la Información en España 2013.* (Ariel, & Planeta, Edits.) Recuperado el 9 de Dic de 2014, de

http://www.fundaciontelefonica.com/arte_cultura/sociedad-de-la-informacion/sie2013/

- *History of robots*. (14 de Dec de 2014). Recuperado el 20 de Dec de 2014, de Wikipedia: https://en.wikipedia.org/wiki/History_of_robots

- *Industrial robot*. (9 de Dec de 2014). Recuperado el 20 de Dec de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Industrial_robot

- *Invention Of the Year: The iPhone*. (01 de Nov de 2007). Recuperado el 11 de Dic de 2014, de Time: http://content.time.com/time/specials/2007/article/0,28804,1677329_1678542,00.html

- *iPhone*. (7 de Dic de 2014). Recuperado el 8 de Dic de 2014, de Wikipedia: https://en.wikipedia.org/wiki/IPhone

- *Manipulator*. (5 de Sept de 2014). Recuperado el 26 de Nov de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Manipulator

- *Military robot*. (21 de Oct de 2014). Recuperado el 26 de Nov de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Military_robot

- *Mobile robot*. (24 de Nov de 2014). Recuperado el 25 de Nov de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Mobile_robot

- Netcraft. (24 de Sept de 2014). *September 2014 Web Server Survey*. Recuperado el 9 de Ene de 2015, de Netcraft: http://news.netcraft.com/archives/2014/09/24/september-2014-web-server-survey.html

- Open Headset Alliance. (5 de Nov de 2007). *Industry Leaders Announce Open Platform for Mobile Devices*. Recuperado el 2014 de Dic de 2014, de http://www.openhandsetalliance.com/press_110507.html

- Ostrovsky, G. (18 de Sep de 2009). *Sensei X Robotic Catheter System for Electrophysiology Procedures*. Recuperado el 27 de Nov de 2014, de http://www.medgadget.com/2009/09/sensei_x_robotic_catheter_system_for_electrophysiology_procedures.html

- Palazzesi, A. (21 de Feb de 2012). *IBM Simon, el primer smartphone de la historia*. Recuperado el 29 de Dic de 2014, de NEOTEO: http://www.neoteo.com/ibm-simon-el-primer-smartphone-de-la-historia/

- *Robot*. (21 de Nov de 2014). Recuperado el 24 de Nov de 2014, de Wikipedia: https://es.wikipedia.org/wiki/Robot

- *Robot*. (4 de Nov de 2014). Recuperado el 24 de Nov de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Robot

- *Smartphone*. (9 de Dec de 2014). Recuperado el 08 de Dic de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Smartphone

- Solomon, B. (20 de Jun de 2013). *Los robots que limpian pisos, detectan bombas y son médicos*. Recuperado el 08 de Dic de 2014, de Forbes México: http://www.forbes.com.mx/los-robots-que-limpian-pisos-detectan-bombas-y-son-medicos/

- *Teleoperation*. (16 de Oct de 2014). Recuperado el 26 de Nov de 2014, de Wikipedia: https://en.wikipedia.org/wiki/Teleoperation

- *Usage Statistics and Market Share of Operating Systems for Websites, January 2015*. (5 de Ene de 2015). Recuperado el 5 de Ene de 2015, de W3Techs: http://w3techs.com/technologies/overview/operating_system/all

- Velasco, J. (28 de Jul de 2011). *El Osborne I, el ordenador "portátil" de 1981*. Recuperado el 3 de Dic de 2014, de hipertextual: http://hipertextual.com/2011/07/el-osborne-i-el-ordenador-portatil-de-1981

- *Wi-Fi*. (10 de Nov de 2014). Recuperado el 10 de Nov de 2014, de Wikipedia: https://es.wikipedia.org/wiki/Wi-Fi